The High Throughput Scheduling Strategy of IHEP

Zou Jiaheng

On behalf of Scheduling Group at IHEP

ISGC2017, 2017.03.10

- IHEP computing environment
- Our scheduling policy with HTCondor
- Supporting systems
- Central Controller
- Summary

IHEP Computing Clusters Overview

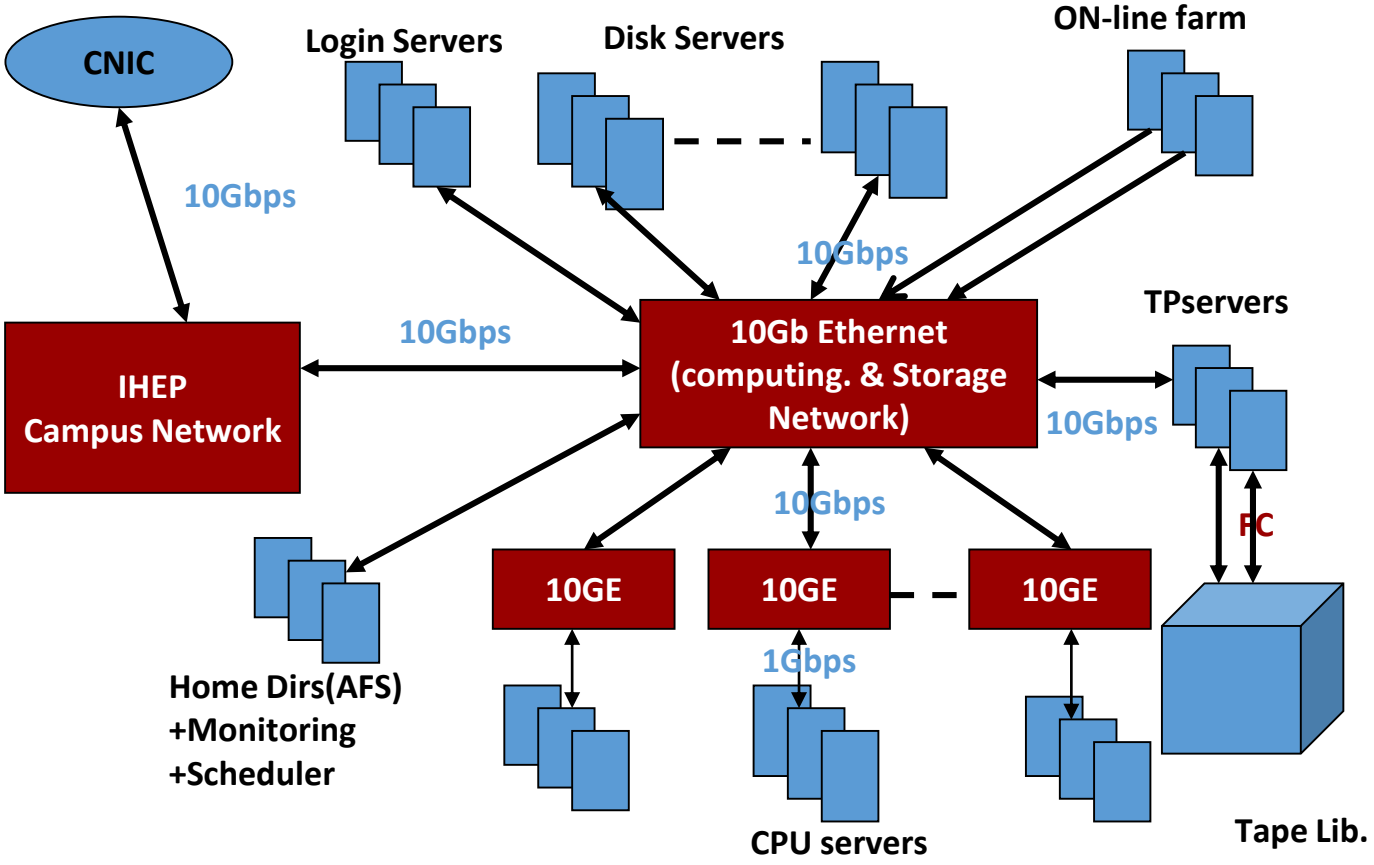
Resources

- ~ 16,000 CPU cores
- ~ 6PB Disk Storage
- ~ 5PB Tape Storage

Users

- BESIII, DYB, JUNO, LHASSO, CMS, ATLAS, etc.
- 1700+ users (~300 active users)
- Up to 100,000+ jobs/day

The Resources Infrastructure



The Migration to HTCondor

PBS was used for more than 10 years

- Limited scalability
- Growing resources and users

Migration to HTCondor

- Better performance for large cluster pool
- Very active community supports
- Step by step with risk control
  - 2015.01 ~ 1100 CPU cores
  - 2016.05 ~ 3500 CPU cores
  - 2016.12 ~ 11000 CPU cores

Current Status of HTCondor

- Architecture
  - 28 submitting nodes
  - 3 scheduler machine (local cluster, virtual cluster, MPI cluster)
  - 3 central manager (local cluster, virtual cluster, MPI cluster)
  - ~ 11000 physical CPU cores + an elastic number of virtual slots
- Jobs
  - Avg 100,000 jobs/day
  - Most are serial single-core jobs
  - A few MPI jobs (currently without statistics)

- IHEP computing environment
- Our scheduling policy with HTCondor
- Supporting systems
- Central Controller
- Summary

Valuable Experience from PBS

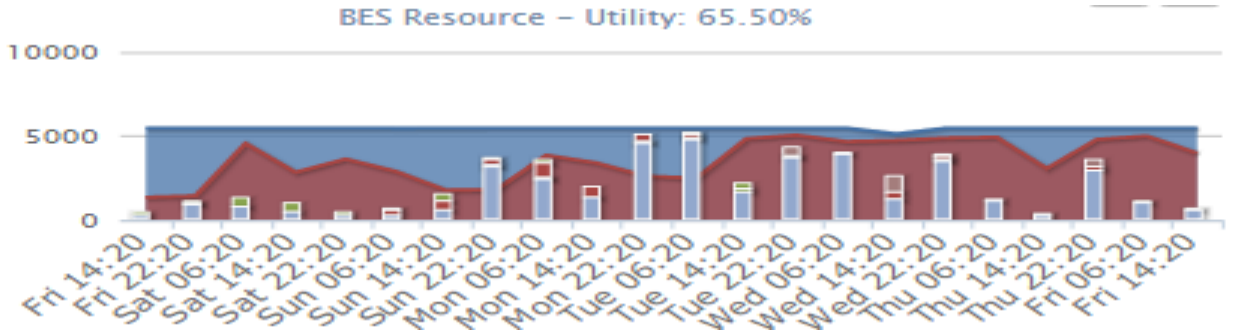
Divided resources and users

- Resources are funded and owned by different HEP Experiments (Groups)
- Users are also grouped by Experiments (Groups)
- There were up to 55 queues with group permission limits in PBS

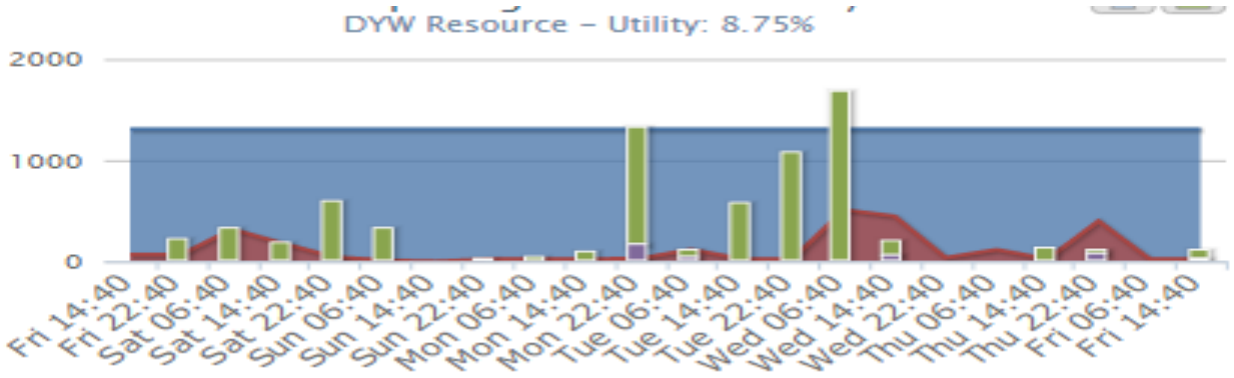
The effects and results

- Coexistence of very busy queues and free queues
- Not very high overall resources utility

Imbalance between Groups



An example of BESIII resource utility



An example of DYB resource utility

The Way of Optimizing

Resource Sharing

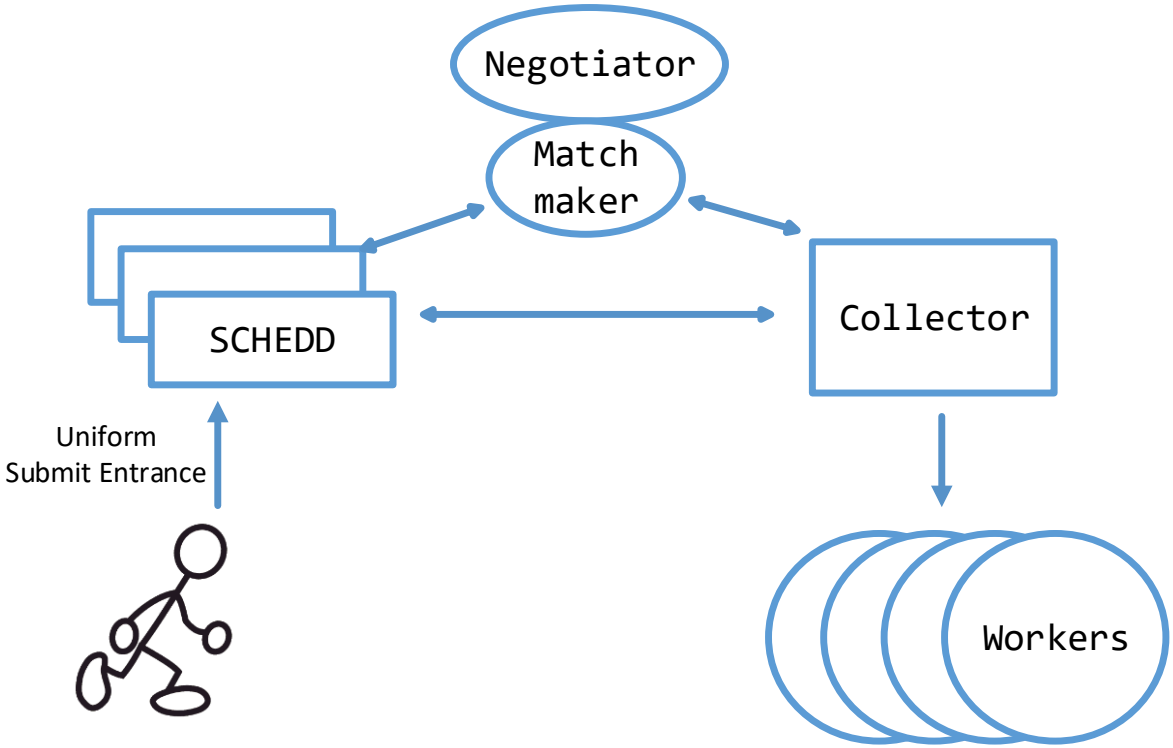
- Break the resource boundary between groups
- Busy groups can take benefits from free groups
  - Busy Group - wants more resource than its own
  - Free Group - less resource allocation than its own

Fairness Ensuring

- The peak resource allocation of different experiments are usually in different time periods
- Free group has higher priority
- The more you share, the more you can allocate

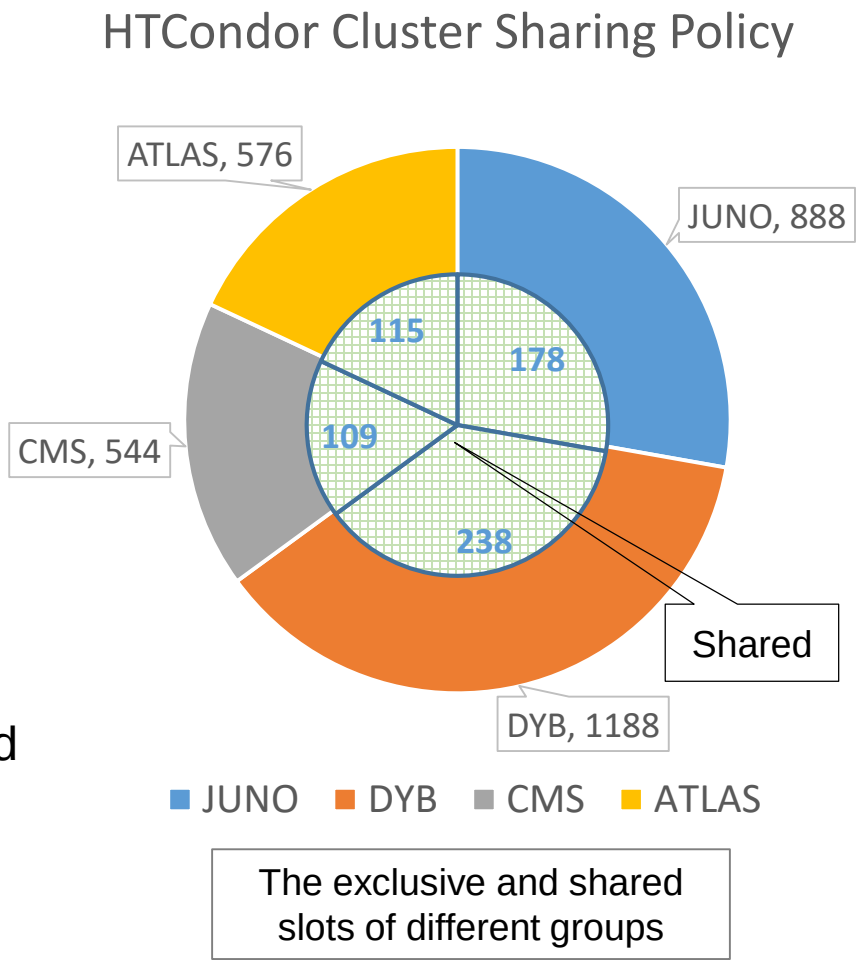
What can we do with HTCondor?

- An efficient MatchMaker for job scheduling
- Simple match rules lead to High Throughput



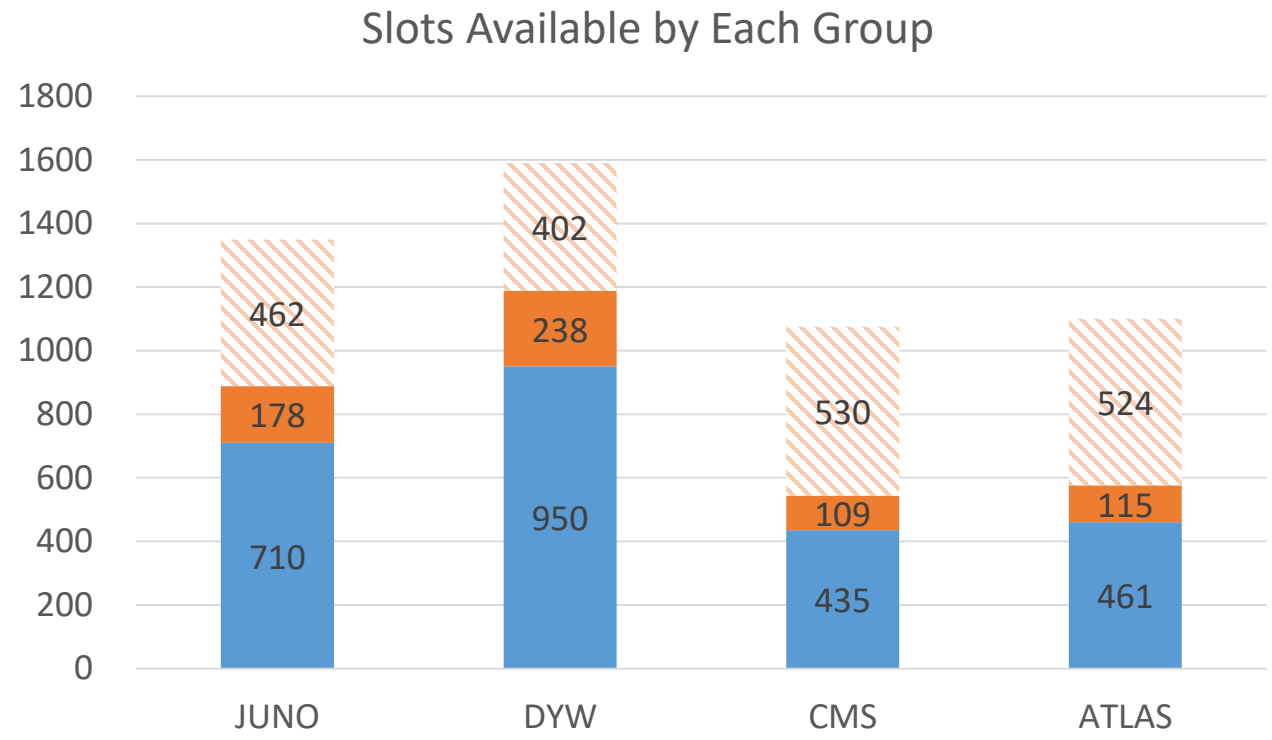
Resource Sharing with HTCondor

- Based on slots (mainly CPU cores)
- As a first step, resources are partially shared
- Exclusive resource
  - Can be matched by owners
- Shared resource
  - Can be matched by all users
  - At least 20% slots are shared by each group
  - It is encouraged to share more by each group



The exclusive and shared slots of different groups

Resource Sharing with HTCondor



The exclusive, shared and max allocable slots for each group
(In May, 2016)

Fairness and Priority

Scheduling preference

- Jobs are preferred to run on exclusive slots
- So that shared slots can be kept for busy groups

Group Quota

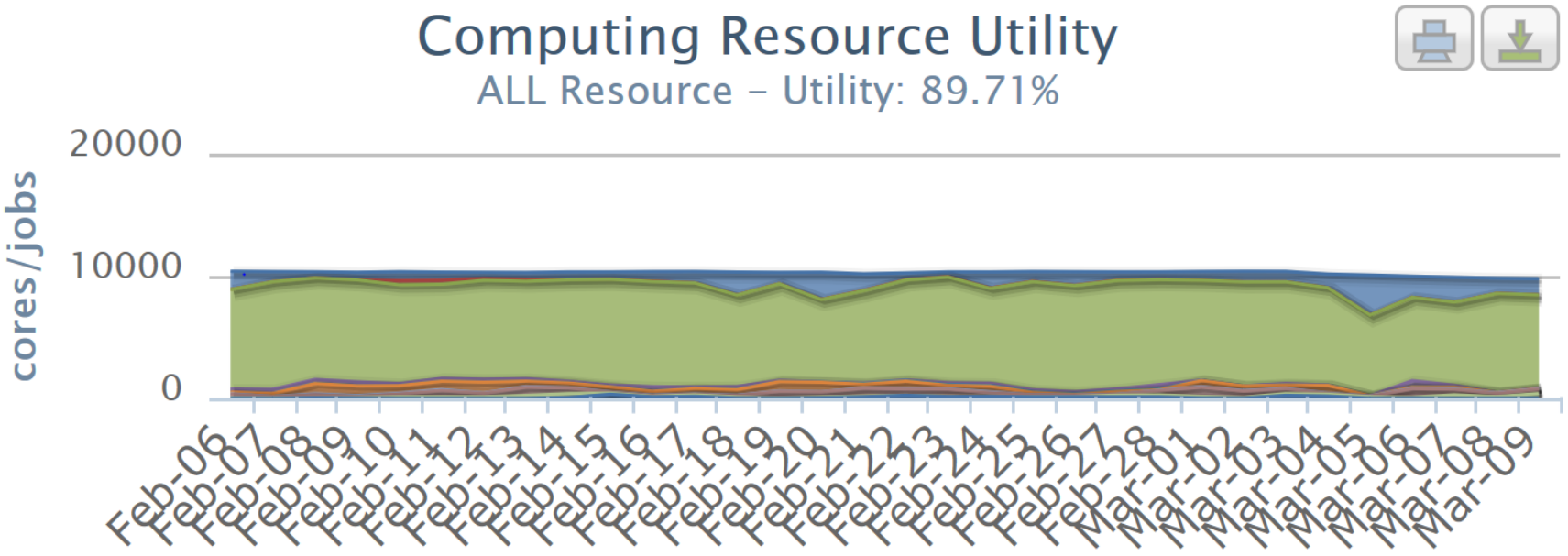
- The initial group quota is set to the real number of owned slots
- Quota can be exceeded if there are shared slots from free groups
- Shared slots are occupied according to the relative ratio of quota between busy groups

Group Priority and User Priority of HTCondor

- Group priority is correlated to the ratio of its occupancy and quota
- User priority is effective in the domain of a group

Resource Utility Improvement

The overall resource utility of last month with HTCondor : ~90%



- The typical resource utility without resource sharing: 50% - 60%
- There is a significant improvement with the resource sharing policy

- IHEP computing environment
- Our scheduling policy with HTCondor
- Supporting systems
- Central Controller
- Summary

The toolkits hepjob

Motivation

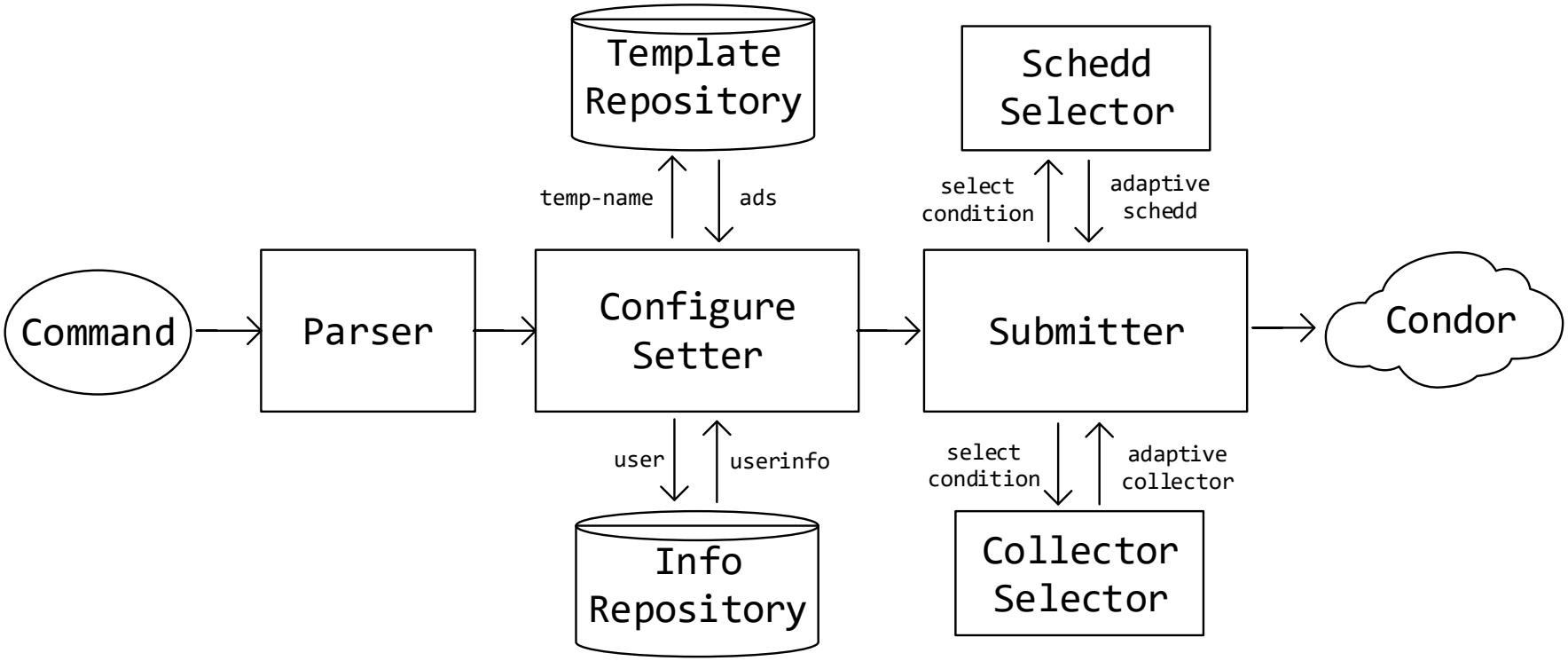
- For users smoothly moving from PBS to HTCondor
- Simple user interfaces
- Help us to achieve our scheduling policy (automatically set additional job attributes)

Implementation

- Based on Python API of HTCondor
- Works with IHEP specific environments
  - Server names, group names …
  - Standard job template for each group/experiment
  - Integration with the IHEP cluster central controller

The toolkit hepjob

The structure of the toolkit

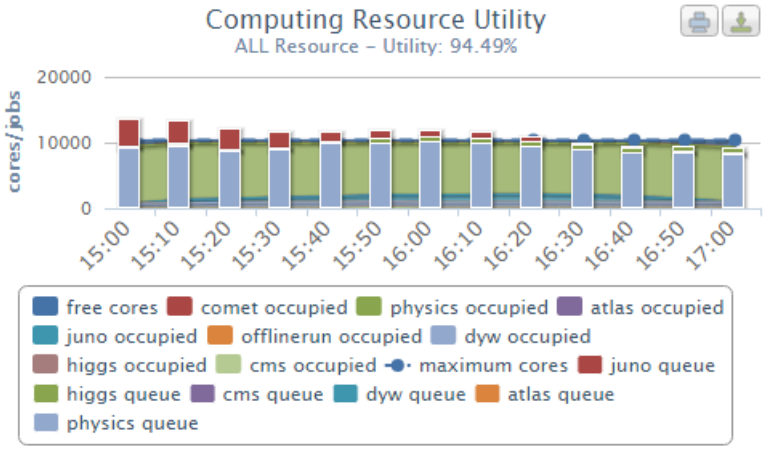


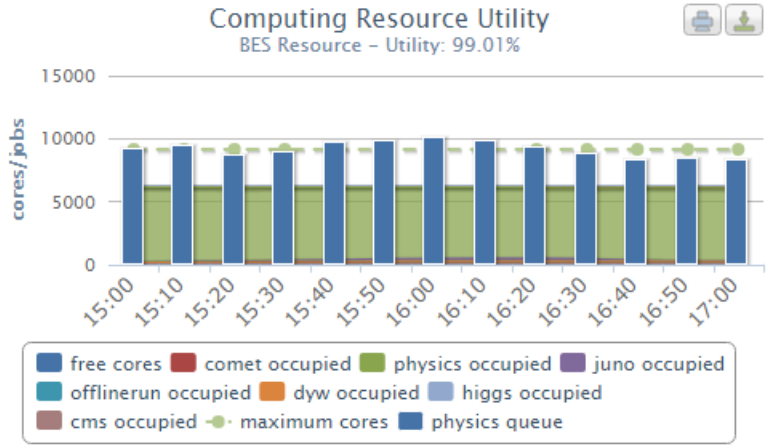
Jobs Monitoring

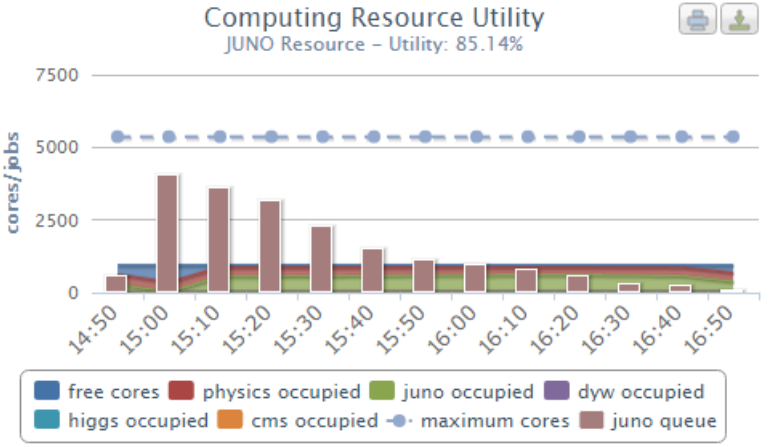
Queueing and running statistics

- The overall clusters
- Each group/experiment

The exclusive and shared resource statistics



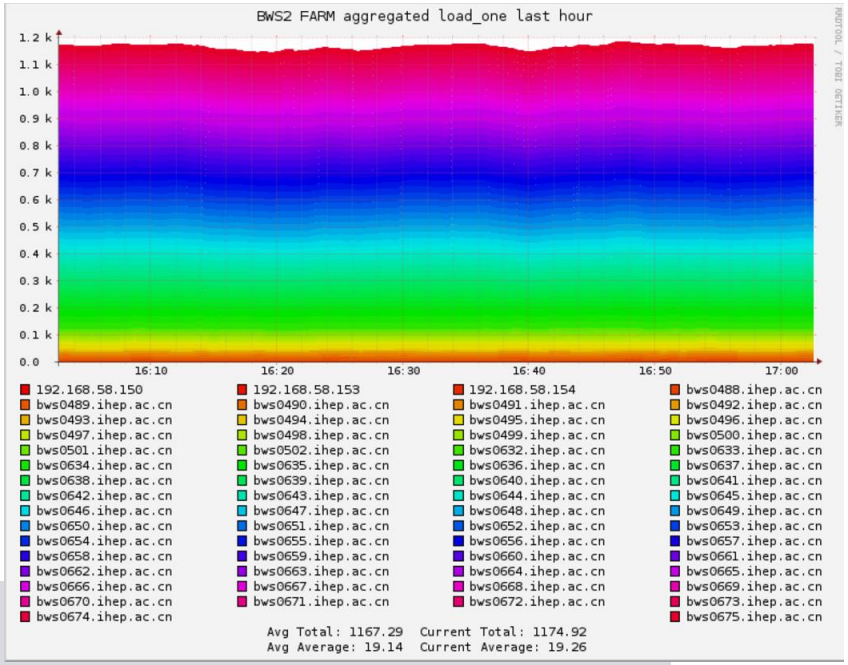


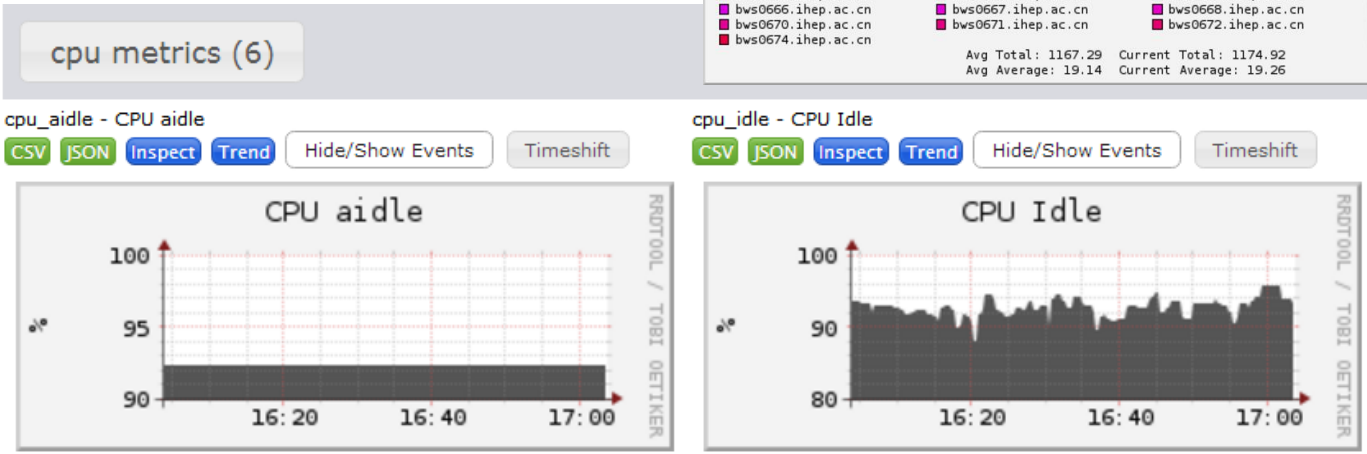


Resources Monitoring

Ganglia

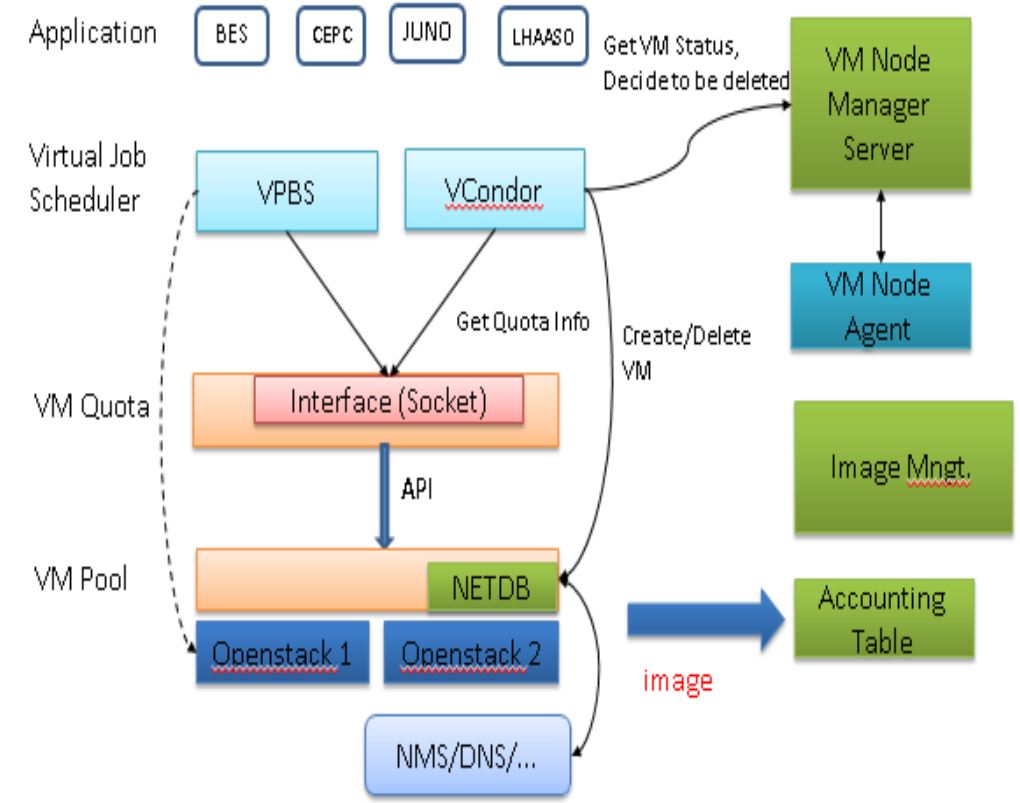
Basic status monitoring for all worker nodes and servers





Virtual Computing

- 28 computing nodes
- 672 CPU cores
- Provide virtual machine on demand of real computing requirement
- Provide an elastic number of slots to busy groups
- Transparent to HTCondor users

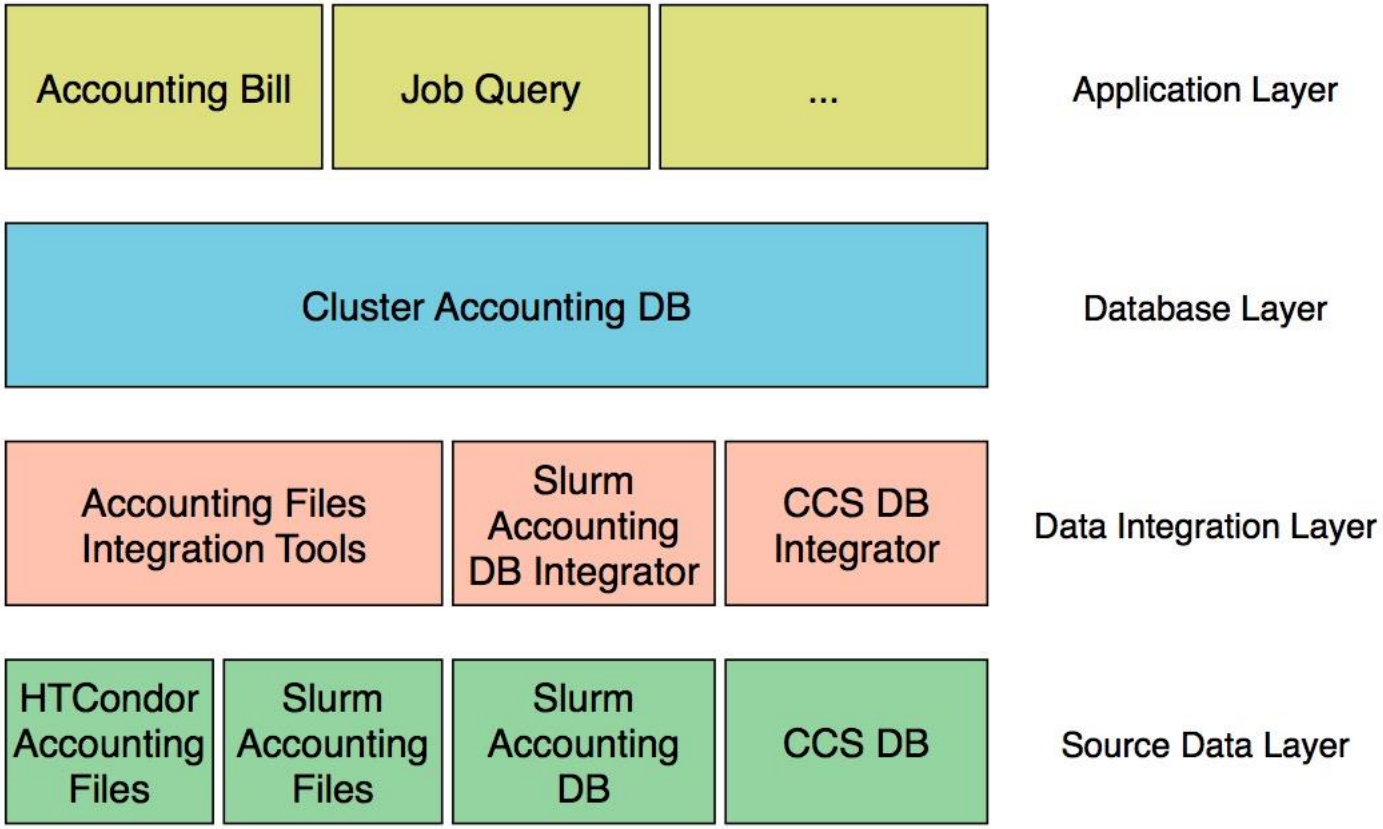


More details in another report :

VCondor – An Implementation of Dynamic Virtual Computing Cluster

Global Accounting

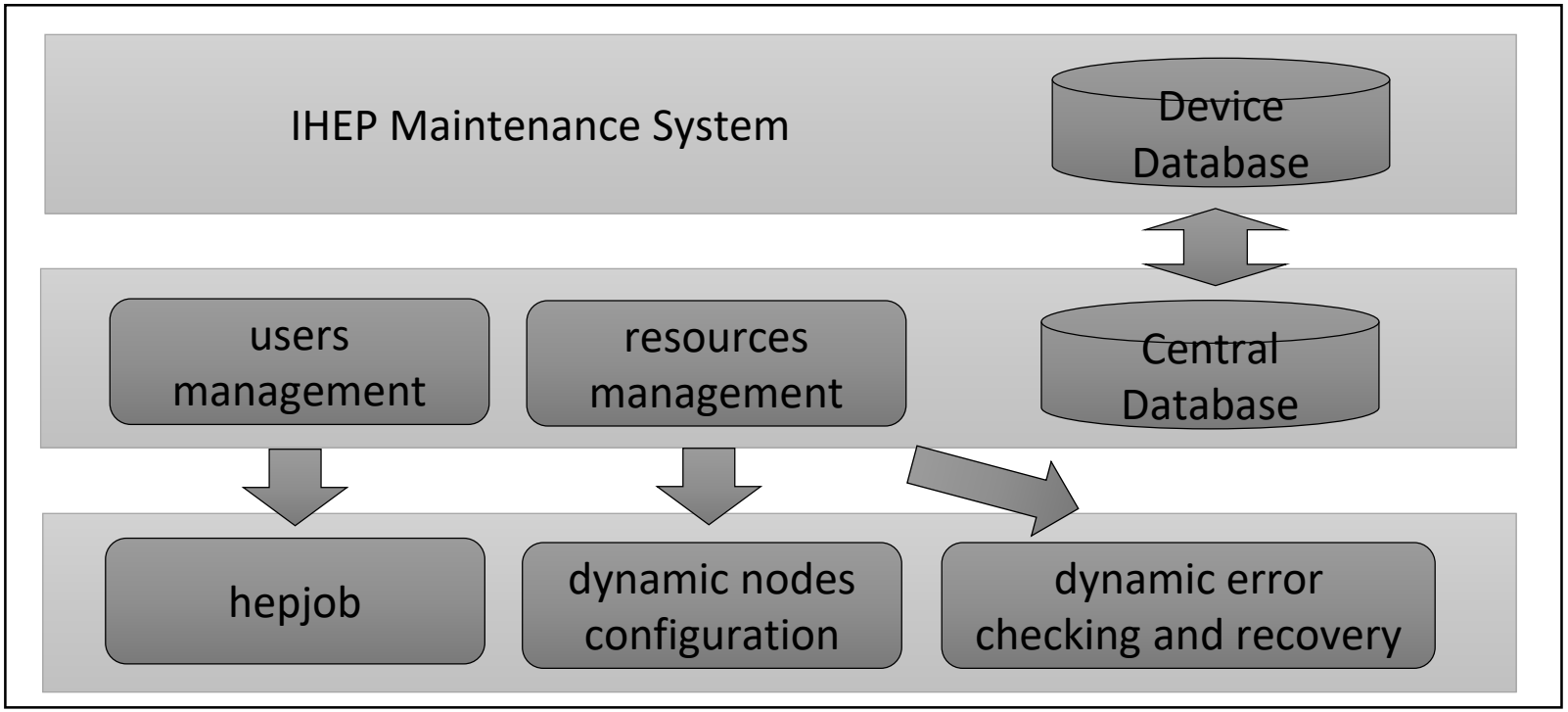
- Detailed accounting to each group and each user
- Weighting slots with slow/fast CPU, Memory, Disk, etc.



- IHEP computing environment
- Our scheduling policy with HTCondor
- Supporting systems
- Central Controller
- Summary

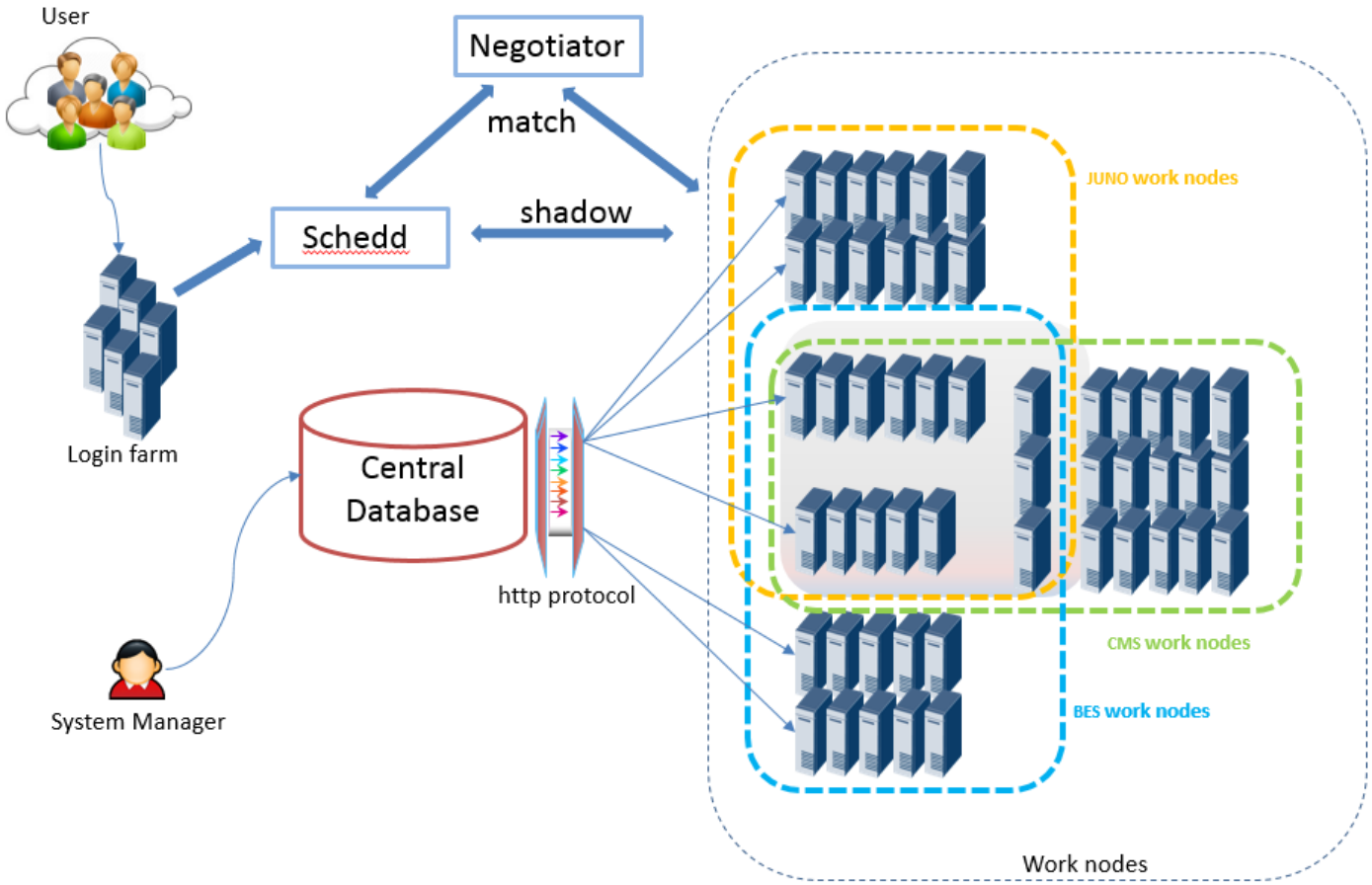
Central Controller System

- The central control of groups, users and resources
  - All information is collected into Central Database
  - Necessary information is published to relative services



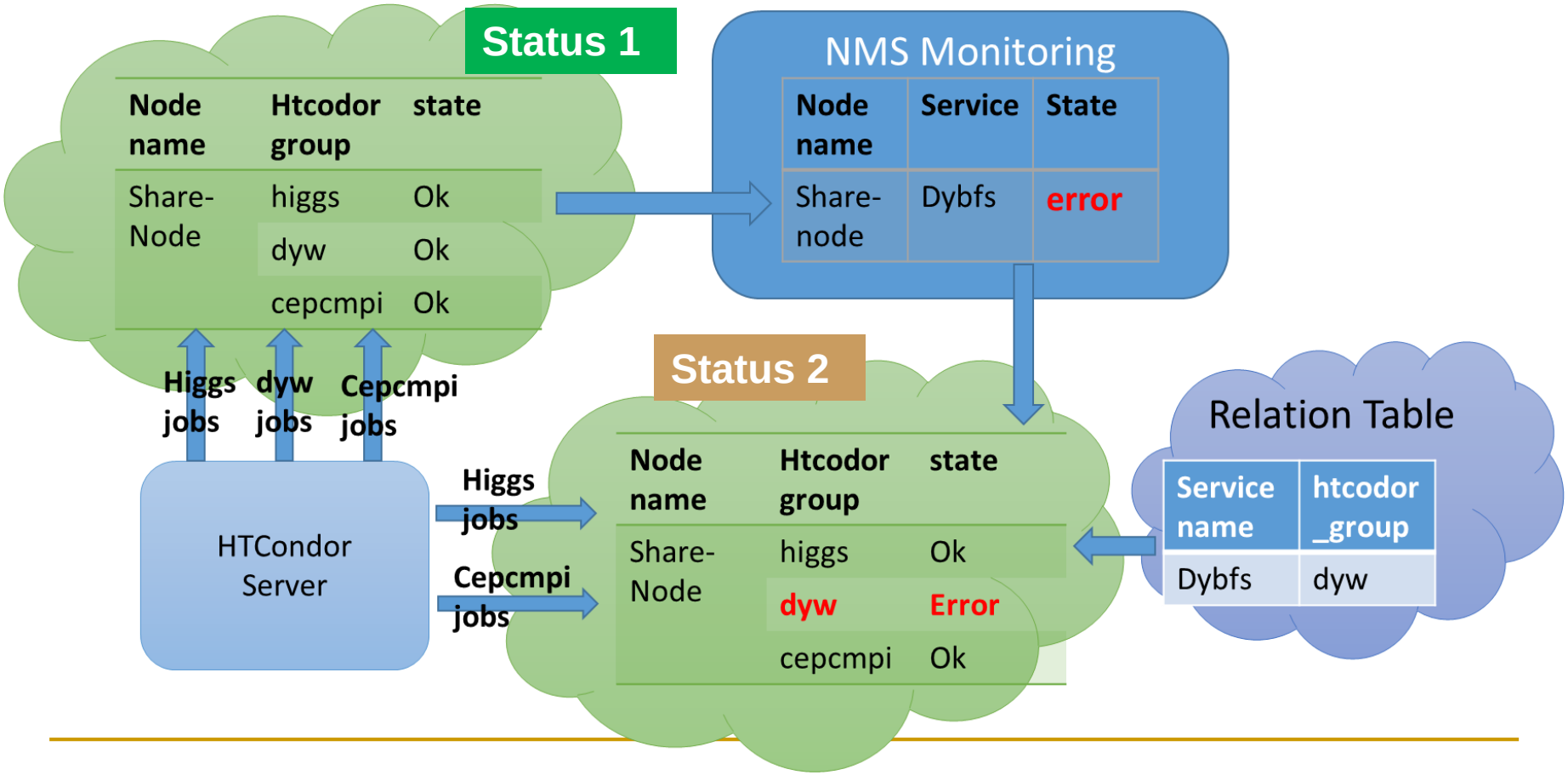
Central Configuration

- The sharing flags (or any attr.) may be changed on a number of nodes
- A hard work manually, but a convenient work via central controller



Error checking and recovery

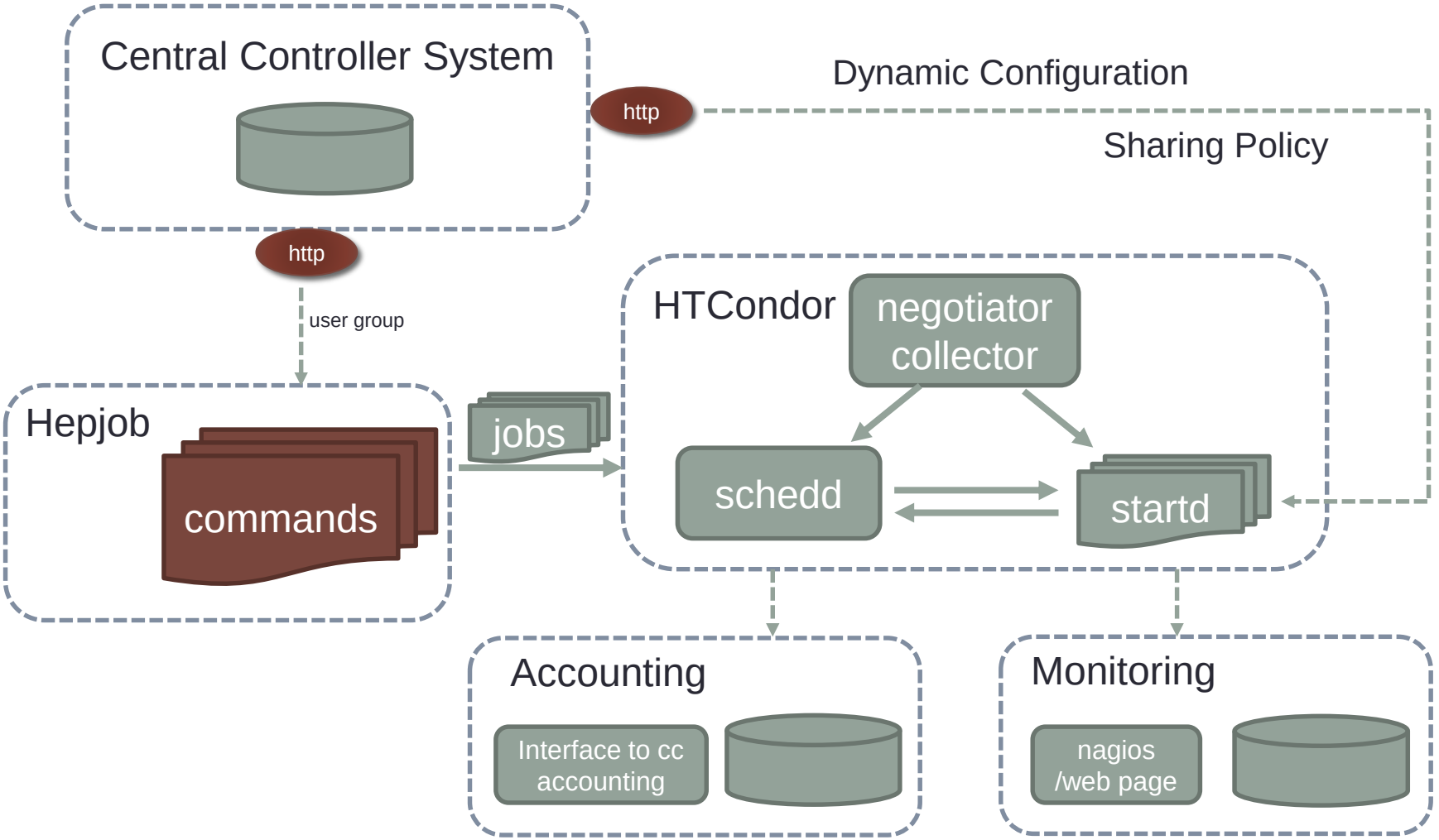
- Health status of all workers are collected into Central Database
- Central controller can modify the workers' attributes automatically



The automatic error checking and recovery mechanism

- IHEP computing environment
- Our scheduling policy with HTCondor
- Supporting systems
- Central Controller
- Summary

Put all together with HTCondor



- Summary
  - The throughput (resource utility) is significantly improved with the resource sharing policy
  - We implemented a number of tools to enhance the system interaction and robustness

- Future Plan
  - Automatically tuning the resource sharing ratio according to the overloads of each group
    - The integration of Job Monitoring and Central Controller
  - Supports for remote HTCondor sites

Thanks!