A Large Ion Collider Experiment

ALICE

# Using the Titan supercomputer for ALICE MC

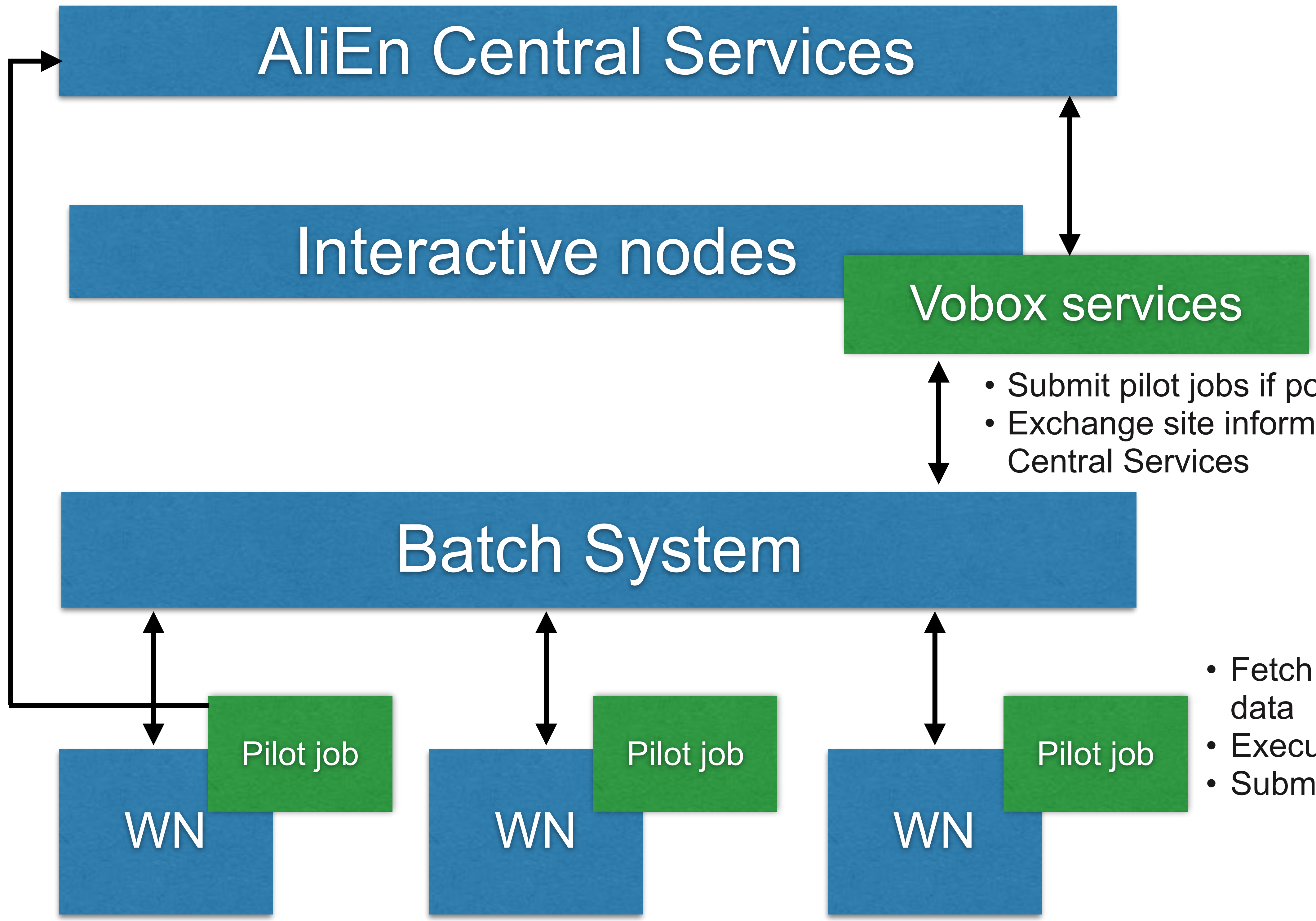
ALICE | BigPanDA@Titan phonecon | Oct 7 2016 | Pavlo Svirin

## Titan general information

| | |
|---|---|
| **Architecture** | 18,688 AMD Opteron 6274 16-core CPUs, 18,688 Nvidia Tesla K20X GPUs |
| **Operating system** | Traditional Linux and Cray Linux Environment (modified SuSE Linux 11) on worker nodes |
| **Memory** | 693.5 TiB (584 TiB CPU and 109.5 TiB GPU) |
| **Disk storage** | 32 PB, 1.4 TB/s IO Lustre filesystem |
| **Peak performance** | 27.1 PF (18,688 compute nodes, 24.5 GPU + 2.6 PF CPU) |
| **I/O Nodes** | 512 service and I/O nodes |

- 2GB RAM/core
- 'Free' resources (in addition to the T2 allocation), potentially up to 10% of the Titan capacity
- Will be used in AliEn environment for Monte-Carlo jobs

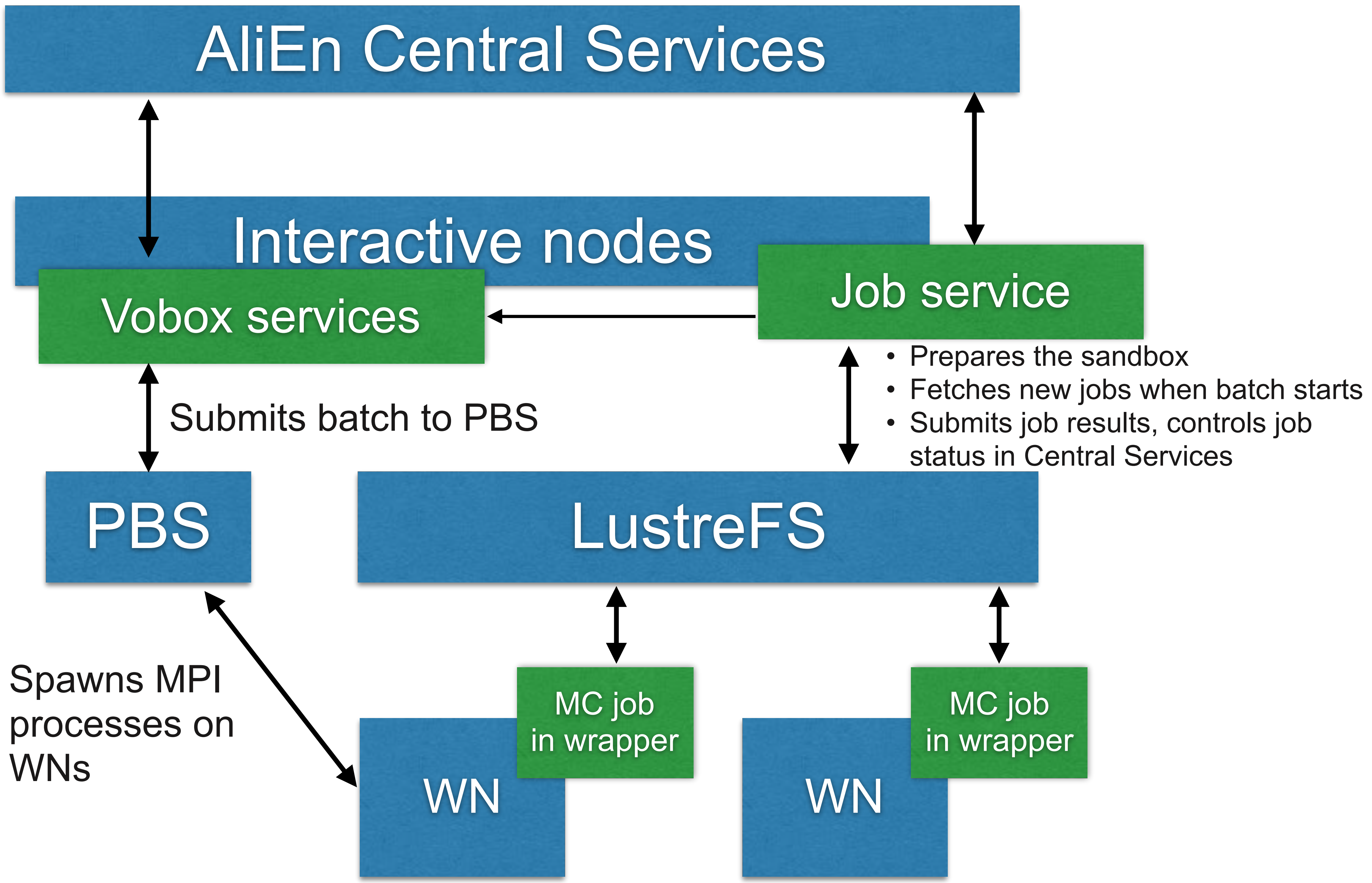
## Usual ALICE Grid Environment elements



- Submit pilot jobs if possible
- Exchange site information with Central Services

- Fetch jobs and its data
- Execute jobs
- Submit job results

## ALICE Grid Infrastructure and ORNL Titan



- Prepares the sandbox
- Fetches new jobs when batch starts
- Submits job results, controls job status in Central Services

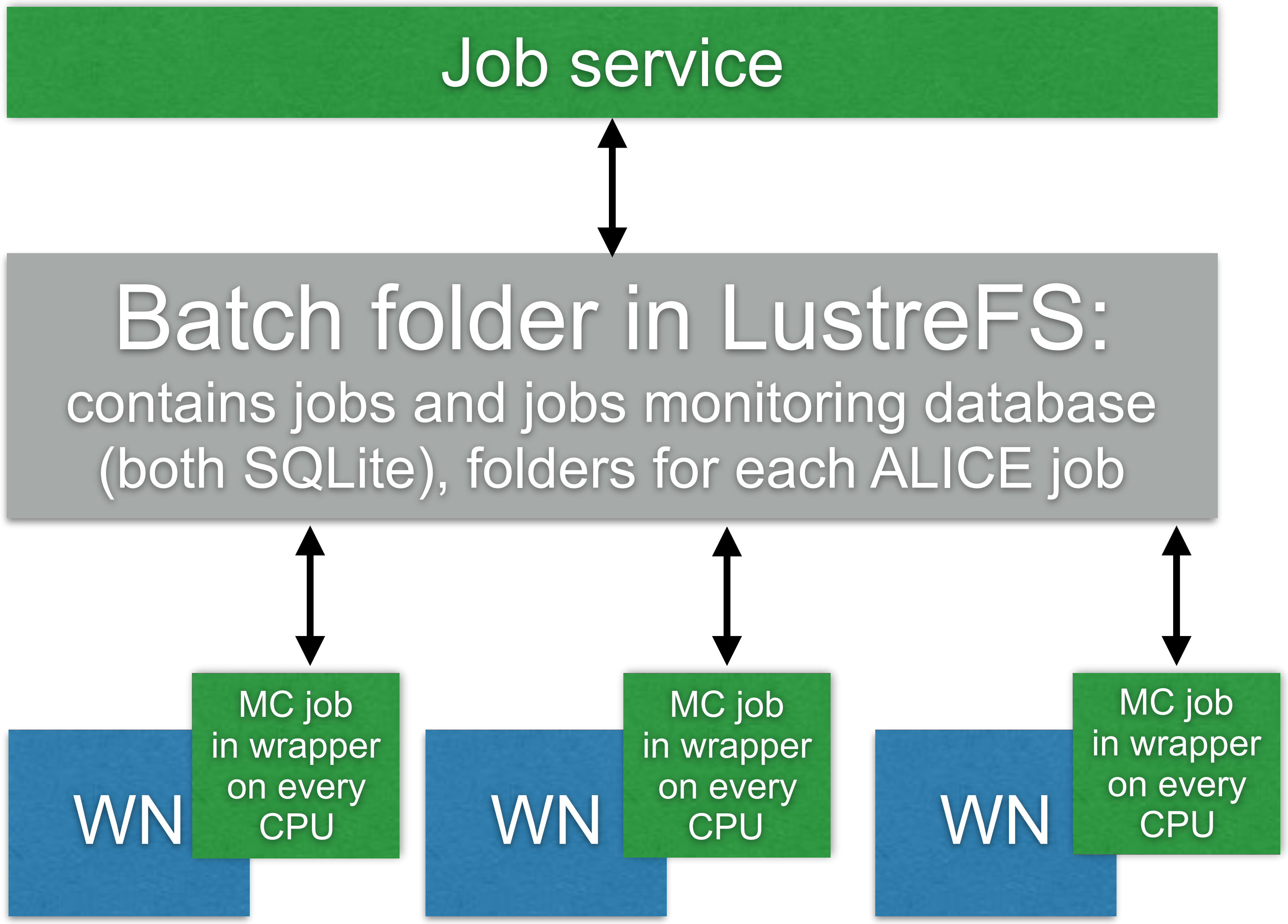
## Batch life cycle

1. **Batch submitted to PBS, waiting**
2. **Batch started**

   creating batch folder, batch info database, waiting for a response from service (second part of the tables are created)
3. **Database structure ready**

   spawning MPI processes, waiting for job information to be available in the database
4. **Job execution**
5. **Cleaning up on exit**

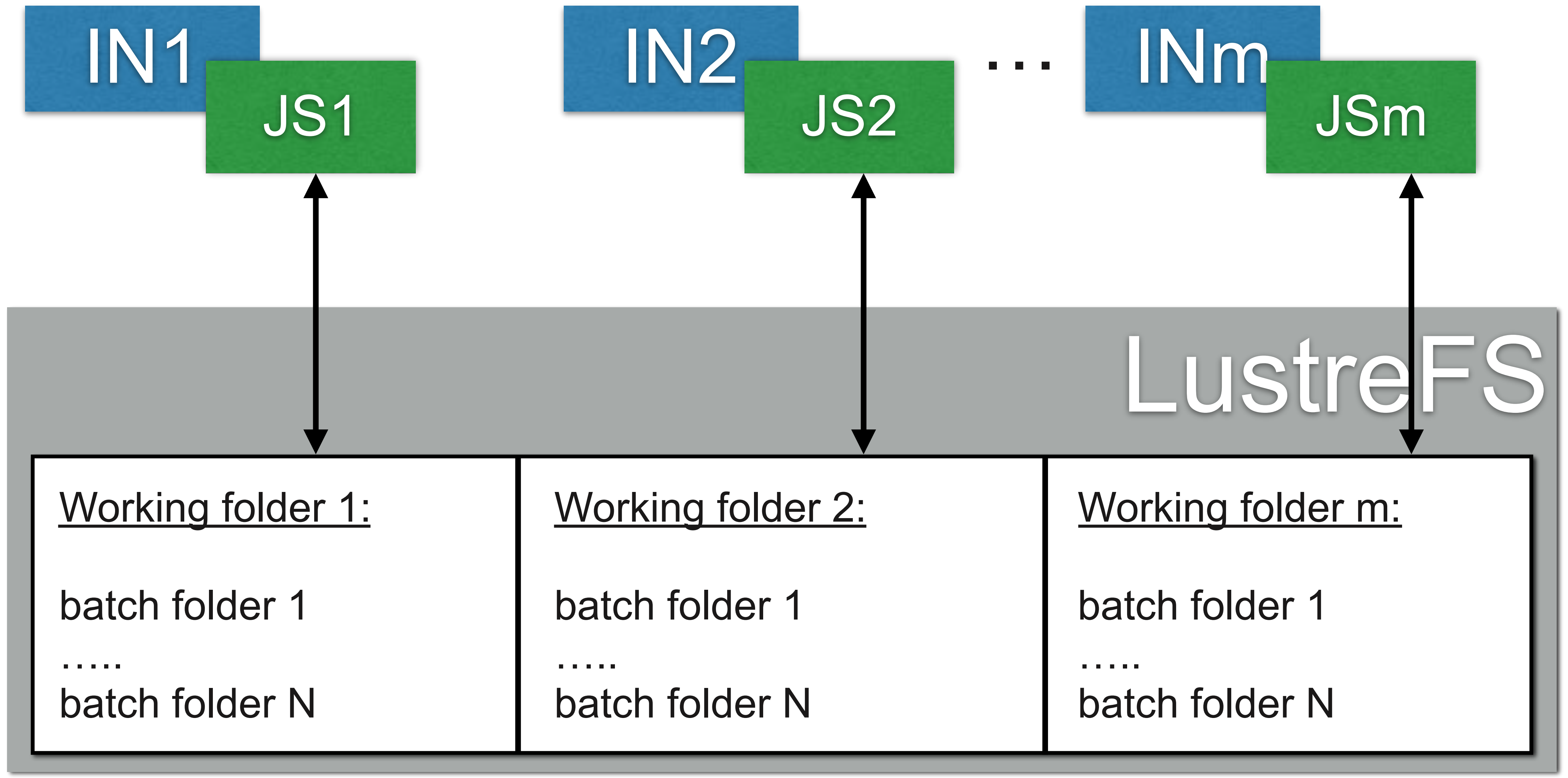
## Titan job service - batch interaction


Job service
Batch folder in LustreFS:
contains jobs and jobs monitoring database (both SQLite), folders for each ALICE job
WN
MC job in wrapper on every CPU
WN
MC job in wrapper on every CPU
WN
MC job in wrapper on every CPU

```
Service_Working_Folder
|_____
|_____2995314
| |_____jalien-job-741337413
| |_____jalien-job-741338229
| |_____jalien-job-741338682
| |_____……
| |_____jobagent.db
| |_____jobagent.db.monitoring
```

```
| |_____jalien-job-741337413
| | |_____ jdl
| | |_____ environment
| | |_____ OCDBsim.root
| | |_____ OCDBrec.root
| | |_____ fifo
| | |_____ aliroot_dpgsim.sh
| | |_____ validation.sh
```

## Possible scaling for Job Services



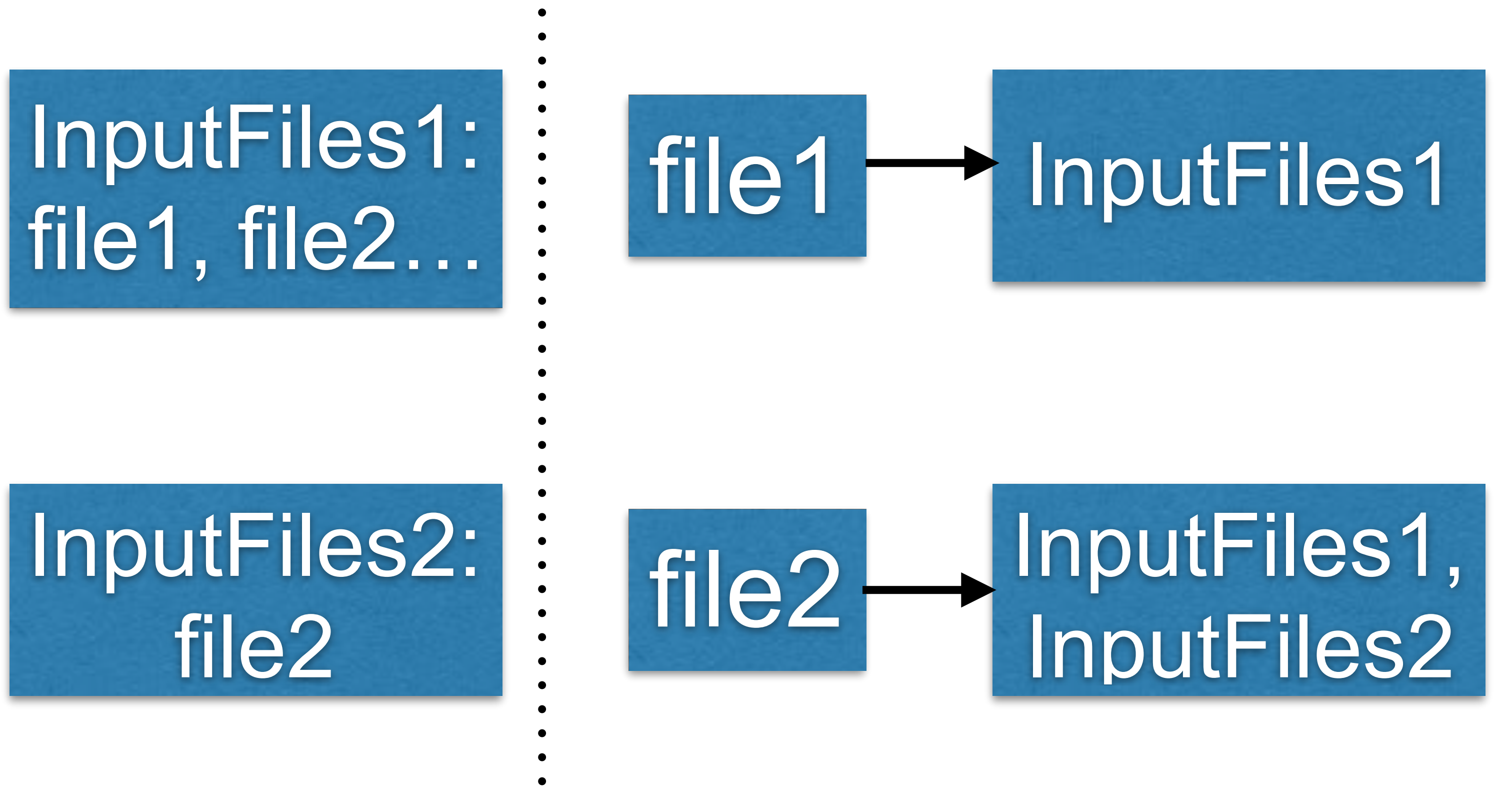
Working folder is selected by Vobox CE service before batch submission (for example, using a simple round-robin)

## Titan job service

- persistent service, originated from JobAgent implementation in Java
- operates with multiple jobs in PBS batches
- does not follow the usual JobAgent workflow completely:
  - spawns multiple connections to Central Services
  - own component for input file download
  - takes completed jobs from the separate stage-out folder when all of the downloads for just started batches finished

## File Download

- job input files are fetched at the same moment of time, possible situations to spawn thousands xrdcp processes
- each file from InputFile element in each JDL fetched is downloaded only once, then it is copied from file cache to job working folder
- can be useful for environments with limited network bandwidth



## Job wrapper

- implemented in Bash
- communicates with the batch databases, reads information about job to be run, reports to the database about job exit codes
- sets the environment variables, starts the executable and validation scripts
- reads resource stats for the job being executed
- provides resource stats in format suitable for sending to AliEn Central Services
- instead of real “uploading” files for done jobs makes hard links into the special “stage-out folder” (takes around 0.5 sec for all of the files)
- can be used with other workflow management systems like PanDa

## ALICE simulation software

- managed to build needed software for Titan (thanks to Dario)
- no special compiler directives were used to build software
- there is a CVMFS repository subset on Titan, updated every hour (mounted under other folder than /cvmfs)
- publisher script had to be brushed up because Titan was cutting too frequent outbound network connections
- now trying to set up publisher script for Cori

## ALICE simulation software challenges

- work on network calls removal succeeded, now the jobs communicate only with snapshots (took a lot of time from July to end of October to fix it for QA and AOD stages)
- network leaks were encountered in QAtrainsim.C and AODtrainsim.C, fixed, merged into master branch
- huge help from Ruben, Roberto and Catalin
- there was a discussion concerning compilation for ACLiC by HLT on validation stage, may be continued in the future

## Software used during tests on Titan

- AliDPG::v5-08-XX-11
- AliDPG::v5-08-XX-13
- GEANT3::v2-1-5
- ROOT:v5-34-30-alice5-1
- AliROOT::v5-08-13h-1
- AliPhysics::v5-08-13h-01-1
- AliPhysics::v5-08-13o-01-1
- jemalloc::v3.6.0

## Running ALICE MC jobs on Titan

- Tried with:

*JobTag = {"comment:Pb-Pb, 5.02 TeV - HIJING min bias - General-purpose Monte Carlo production anchored to Pb-Pb 5.02 TeV runs (LHC15o), ALIROOT-6784"};*

processing takes up to 3 hours for 1 event

- LHCbMarks: 5.56 for worker node CPU, corresponds to estimated 0.35/events per hour, 7.60 on interactive nodes
- with Pb-Pb jobs we can not participate in pure backfill (usually less than 2 hours), CSC108 project has the lowest priority, thus, Titan tools do not show accurate info about start time
- quite successful in requesting 125 nodes/5:45h slots which can be ok for 2 events (theoretically up to 10 slots per day)
- does not seem too problematic to get 125 nodes/3:30 slots (max waiting time observed - around 3 hours)
- more suitable for running p-p MC jobs (no benchmarks yet due to a problem on simulation stage)

## Overhead estimation for starting 2000 jobs from the same masterjob

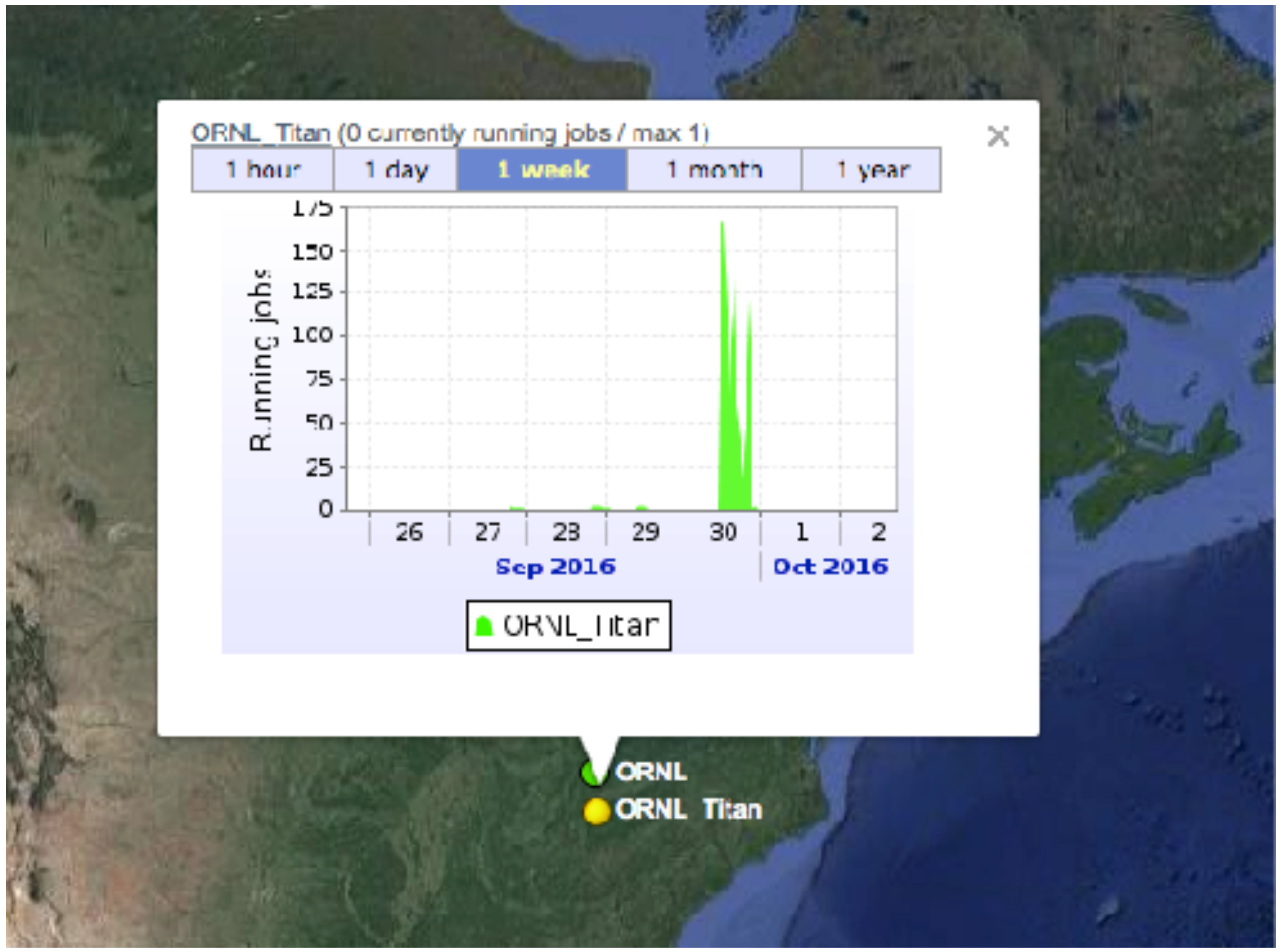
- Downloading 2 snapshot files around 130Mb: ~2 mins (downloads run in parallel)
- Copying 2 snapshot files to job working folder on LustreFS: ~0.4 sec
- Fetching 2000 JDLs: ~4 mins
- **Total time (worst case) : 4*60 + 2*60 + 0.4*2000 = ~1160 sec = ~20 mins**
- Making hard link for a 2 files instead of copying: 0.006 sec
- If we use recommendation for LustreFS "1 file per 1 node":
- **Total time: 4*60 + 2*60 + 0.4*2000/16 + 0.006*2000 = ~422 sec = 7 mins**

**Overhead can be optimized by:**

- using local storage element for xrootd (e.g. on CADES) or running service on a DTN node
- extending JDL fetching from central queue from "give me 1 job" to, e.g., "give me 50 jobs"

# Titan in ALICE monitoring system

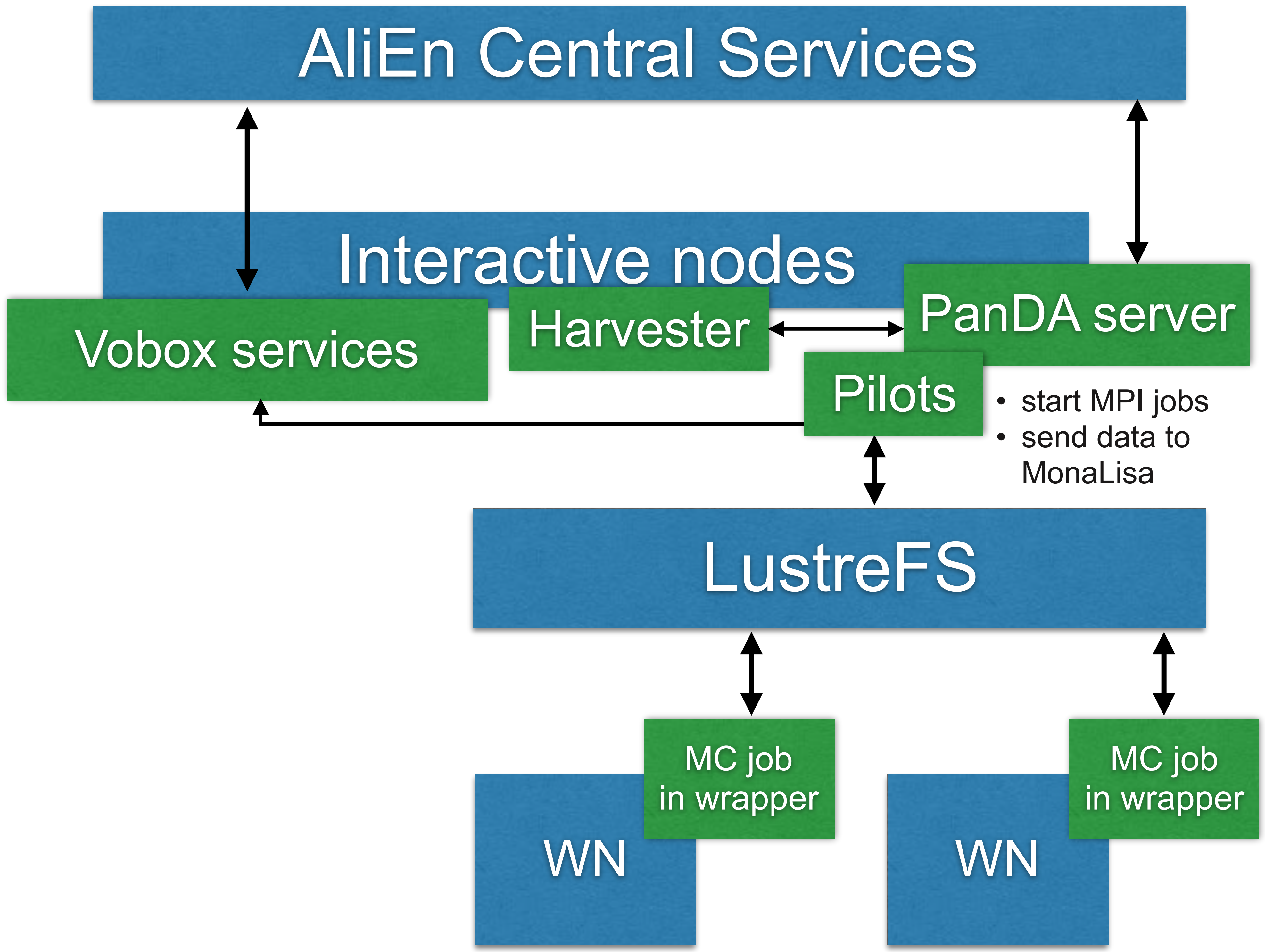
ORNL
ORNL Titan
UNAM T1
UNAM



## jAliEn for Titan status

- tested with real payloads for the first time
- showed stable operation
- some jobs (1-3%) go to ERROR_ASSIGNED state while fetching big number of JDLs (2000), needs to be investigated
- alimonitor understands RUNNING state for the jobs but does see DONE/ERROR_* states, has to be fixed
- X509 proxy has to be introduced into jAliEn
- fetching multiple JDLs within one request can be simulated with a HTTP/JSON request to a Tomcat which can translate it into jAliEn entities (just a suggestion so far)

## PanDA integration and Pilot2 (draft design)



- start MPI jobs
- send data to MonaLisa

- Fetches jobs from Central job queue, uploads finished jobs
- Fetches pilot descriptions from Harvester service
- Runs pilots

## PanDA integration: details and challenges

- PanDA server takes pilot description from Harvester service (more: https://indico.cern.ch/event/526308/contributions/2247704/attachments/1318598/1976659/Harvester.pdf )
- uses pre-binding for jobs: jobs need to be kept in ASSIGNED state for a certain period
- possible to play with "—mode" job option to split the job stages between the time slots (has to be tested)
- we can use HTTP/JSON calls for running jAliEn commands through Tomcat (approach has been tested in August 2015)
- bash job wrapper ready for PanDA

## Conclusions and future work

- a new generation of ALICE Grid software running on Titan (however, needs more testing)
- Titan marked as functioning in our monitoring system (more testing needed)
- we managed to remove network calls from simulation software
- takes a long time to process Pb-Pb jobs, benchmark needed for p-p
- payload JDLs have to be discussed taking into account the benchmarks, current and future data specifics
- started a closer interaction with PanDA team

# THANK YOU!