Job splitting on the ALICE grid, introducing the new job optimizer for the ALICE grid middleware.

Haakon André Reme-Ness (Western Norway University of Applied Sciences) on behalf of the ALICE collaboration.

Introduction

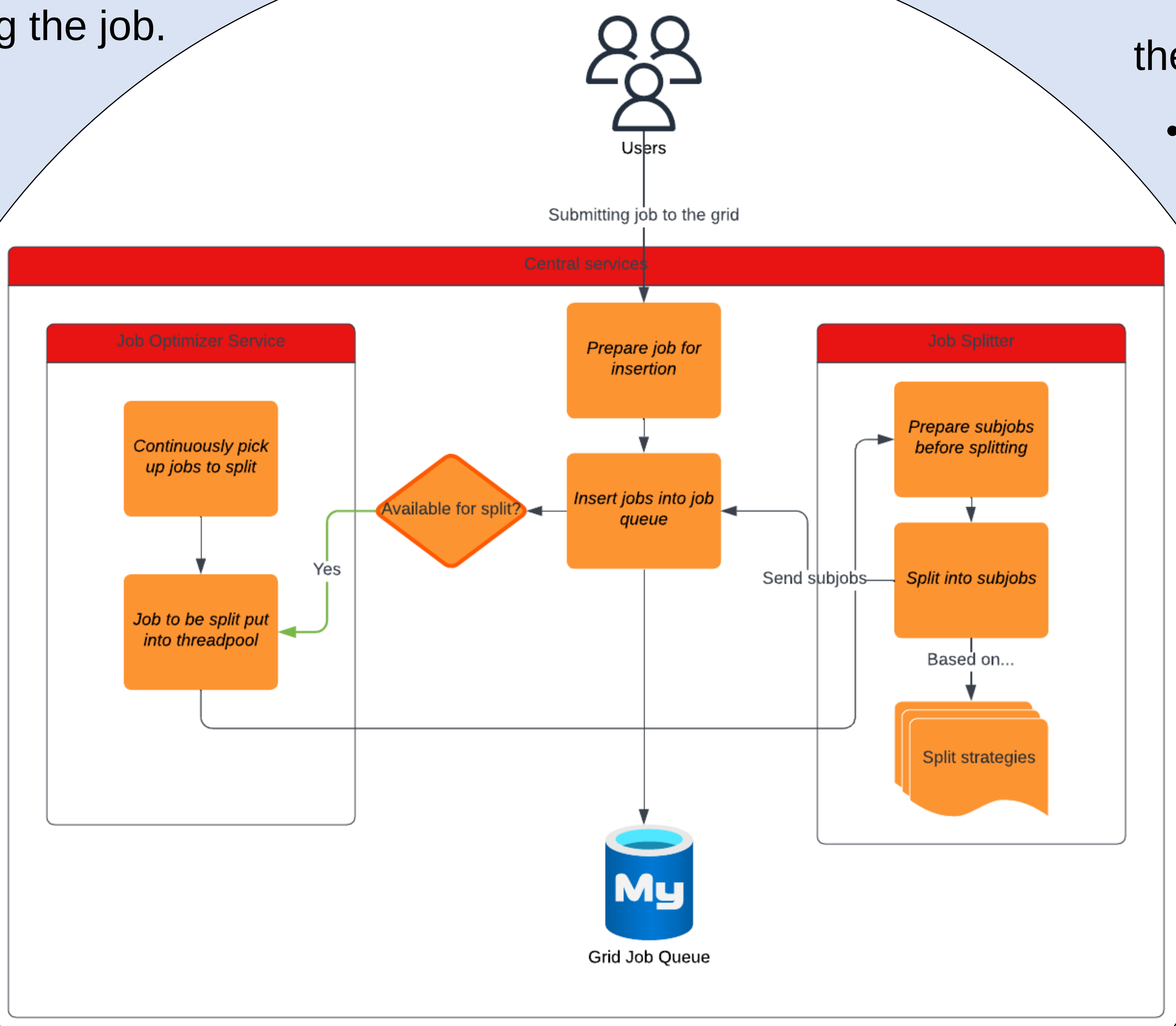
The ALICE experiment at the CERN LHC has undergone a significant upgrade of the detectors, readout, and software prior to Run 3 (2022 - onward). Following the upgrades, ALICE will collect, reconstruct and analyze approximately 10x more events than in the previous data-taking period. In preparation for the increased requirements for the distributed computing system, ALICE has developed and deployed a new Grid middleware JAliEn, which adopted the functionality and updates accumulated in the past 15 years. It makes use of new software tools and modern development practices. A critical part of the payload management of JAliEn is the so-called Job Optimizer. Based on a general job submitted by a user the Job Optimizer prepares a specific set of sub-jobs compatible with the site resources, in particular with the data location, software requirements, quotas, and priorities. The newly developed Job Optimizer is presented in this poster.

Submitting a job to the grid

- User submits a job based on a JDL (Job Description Language) file.
- Evaluate and validate the JDL.
- Prepare requirements for job, such as ensuring required packages are available on sites executing the job.
- Insert original job into job queue .
- Not ready to be picked up by a site yet.
- If available, try to perform job split.
- If not, the job gets picked up by the Job Optimizer later.

Job Optimizer service

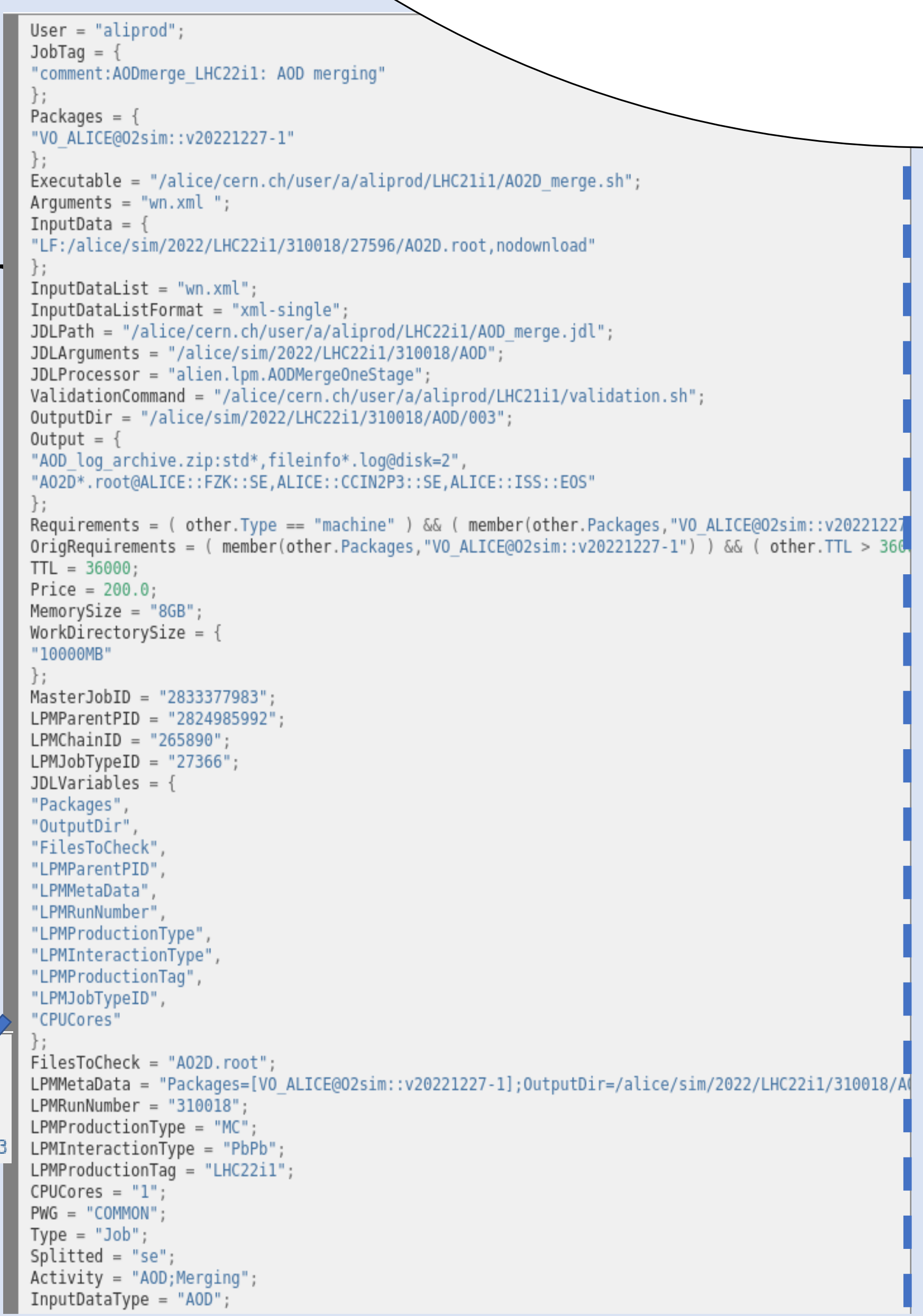
- Continuously running with a short cooldown period.
- Picks up job ready to be split from the job queue, based on how old the job is.
- Submit job id to a thread-pool that starts the job splitting.
  - Size of the thread-pool determines how many jobs a machine can split at once and is a configurable parameter for central machines to assists with scaling.


Users
Submitting job to the grid
Central services
Job Optimizer Service
Continuously pick up jobs to split
Job to be split put into threadpool
Yes
Available for split?
Prepare job for insertion
Insert jobs into job queue
Job Splitter
Prepare subjobs before splitting
Split into subjobs
Send subjobs
Based on...
Split strategies
Grid Job Queue

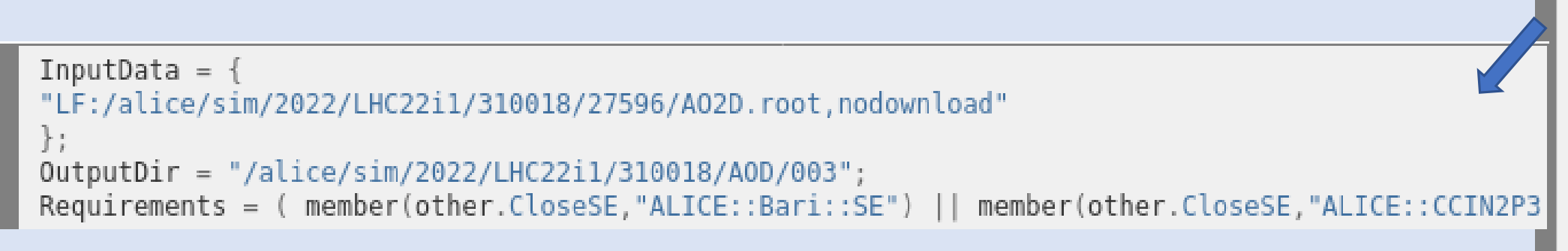
Overview of a rough workflow for the job optimizer.

Database Optimization

- Optimization towards database is mainly introduced with the new Job Optimizer.
- Inserting sub-jobs is now done as a transaction, all jobs are inserted or none.
- Checking for datafile physical location done in bulks, making use of the partitioning of tables.
- Update and Select in one query when picking jobs to split, not ideal in MySQL, but possible with user-defined variables.
- Describing sub-jobs in the job queue database as the difference from master-job, redundant information.



```
User = "aliprod";
JobTag = {
"comment:AODmerge_LHC22i1: AOD merging"
};
Packages = {
"VO_ALICE@O2sim::v20221227-1"
};
Executable = "/alice/cern.ch/user/a/aliprod/LHC21i1/AO2D_merge.sh";
Arguments = "wn.xml ";
InputData = {
"LF:/alice/sim/2022/LHC22i1/310018/27596/AO2D.root,nodownload"
};
InputDataList = "wn.xml";
InputDataListFormat = "xml-single";
JDLPath = "/alice/cern.ch/user/a/aliprod/LHC22i1/AOD_merge.jdl";
JDLArguments = "/alice/sim/2022/LHC22i1/310018/AOD";
JDLProcessor = "alien.lpm.AODMergeOneStage";
ValidationCommand = "/alice/cern.ch/user/a/aliprod/LHC21i1/validation.sh";
OutputDir = "/alice/sim/2022/LHC22i1/310018/AOD/003";
Output = {
"AOD_log_archive.zip:std*,fileinfo*.log@disk=2",
"AO2D*.root@ALICE::FZK::SE,ALICE::CCIN2P3::SE,ALICE::ISS::EOS"
};
Requirements = ( other.Type == "machine" ) && ( member(other.Packages,"VO_ALICE@O2sim::v2022122
OrigRequirements = ( member(other.Packages,"VO_ALICE@O2sim::v20221227-1") ) && ( other.TTL > 360
TTL = 36000;
Price = 200.0;
MemorySize = "8GB";
WorkDirectorySize = {
"10000MB"
};
MasterJobID = "2833377983";
LPMParentPID = "2824985992";
LPMChainID = "265890";
LPMJobTypeID = "27366";
JDLVariables = {
"Packages",
"OutputDir",
"FilesToCheck",
"LPMParentPID",
"LPMMetaData",
"LPMRunNumber",
"LPMProductionType",
"LPMInteractionType",
"LPMProductionTag",
"LPMJobTypeID",
"CPUCores"
};
FilesToCheck = "AO2D.root";
LPMMetaData = "Packages=[VO_ALICE@O2sim::v20221227-1];OutputDir=/alice/sim/2022/LHC22i1/310018/A
LPMRunNumber = "310018";
LPMProductionType = "MC";
LPMInteractionType = "PbPb";
LPMProductionTag = "LHC22i1";
CPUCores = "1";
PWG = "COMMON";
Type = "Job";
Splitted = "se";
Activity = "AOD;Merging";
InputDataType = "AOD";
```



```
InputData = {
"LF:/alice/sim/2022/LHC22i1/310018/27596/AO2D.root,nodownload"
};
OutputDir = "/alice/sim/2022/LHC22i1/310018/AOD/003";
Requirements = ( member(other.CloseSE,"ALICE::Bari::SE") || member(other.CloseSE,"ALICE::CCIN2P3
```

Example of how much information is redundant for a sub-job when using a full JDL.

Job splitter

- Splitting is done by splitting up the data files to different sub-jobs.
- Several splitting strategies, split based on data locality being one.
- Splitting based on data locality is more resource demanding as queries against databases to find physical location must be done.
- Splitting based on locality might also trigger merging of sub-jobs, as some sub-jobs might contain too few datafiles.
- A user must set a maximum threshold for number of datafiles per sub-job, and this parameter is used to also get the minimum if not defined by user.
- Second major job splitting algorithm is aimed at Monte-Carlo simulation payload, where the difference is the random seed for the MC and output directory per sub-job.

26TH INTERNATIONAL CONFERENCE
CHEP2023
Computing in High Energy & Nuclear Physics

Poster #220

Western Norway University of Applied Sciences
ALICE