Western Norway University of Applied Sciences

ALICE

Job Optimizer for JAliEn

Haakon André Reme-Ness

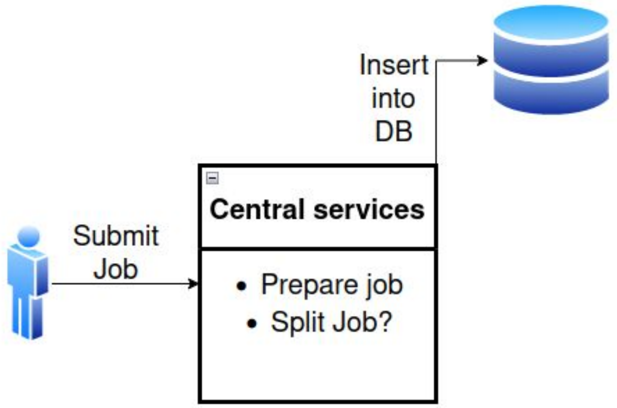
ALICE Tier-1/Tier-2 Workshop, Budapest 26.09.2022

Background

- New job optimizer and job splitter should not be different for users
  - Old JDL should work the same way
  - Some new arguments to the JDL added
- More focus on performance
  - Try balance loads better between central service machines
  - As little load as possible on the database
  - Hopefully faster splitting

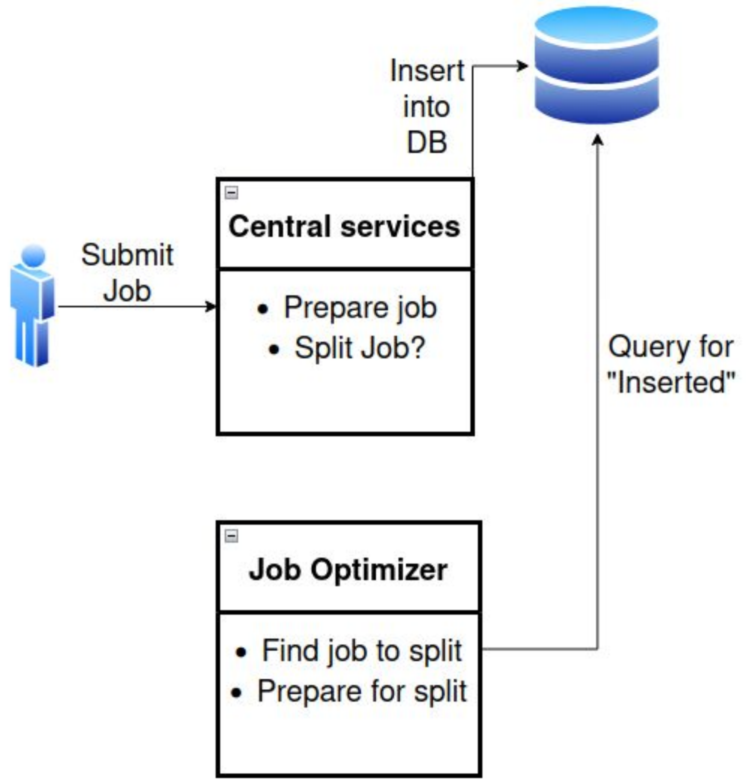
Workflow user submit

- User submit job with JDL
- JDL will be checked to ensure it is correct
- Check for user quota
- Prepare JDL and job to be inserted
- If job is not be split, insert into Queue database as **Waiting**
    - Together with corresponding jobagent to match with site
- If splitting, insert into database as **Inserted**
    - Split is defined in JDL



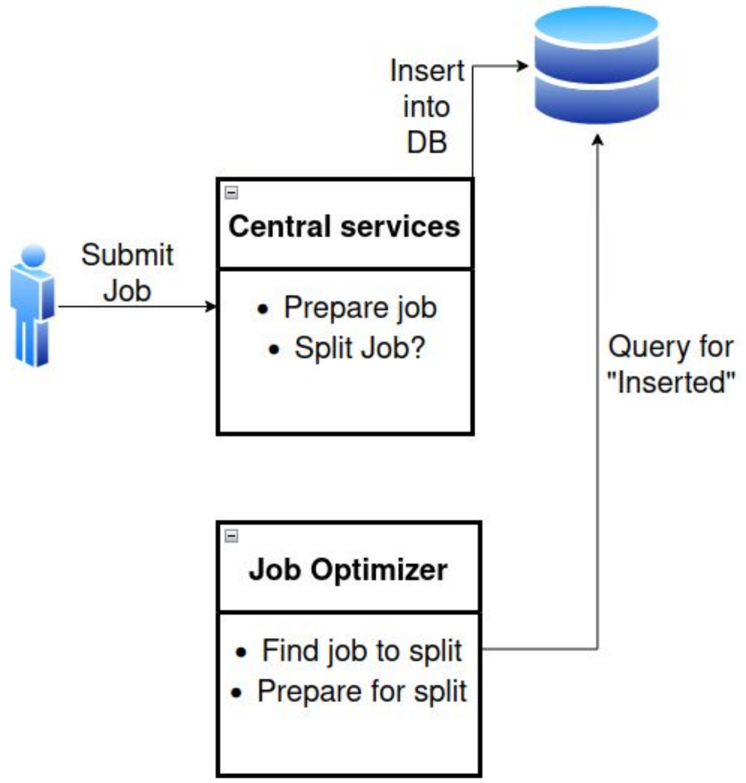
Workflow Job Optimizer

- Job optimizer is a continually running service on central service machines
- Local queue with max 5 threads
  - Each thread will try to pick a job to split
  - Does an update on oldest splitjob with status **Inserted**, setting status to **Splitting**
- Send the job to the job splitter



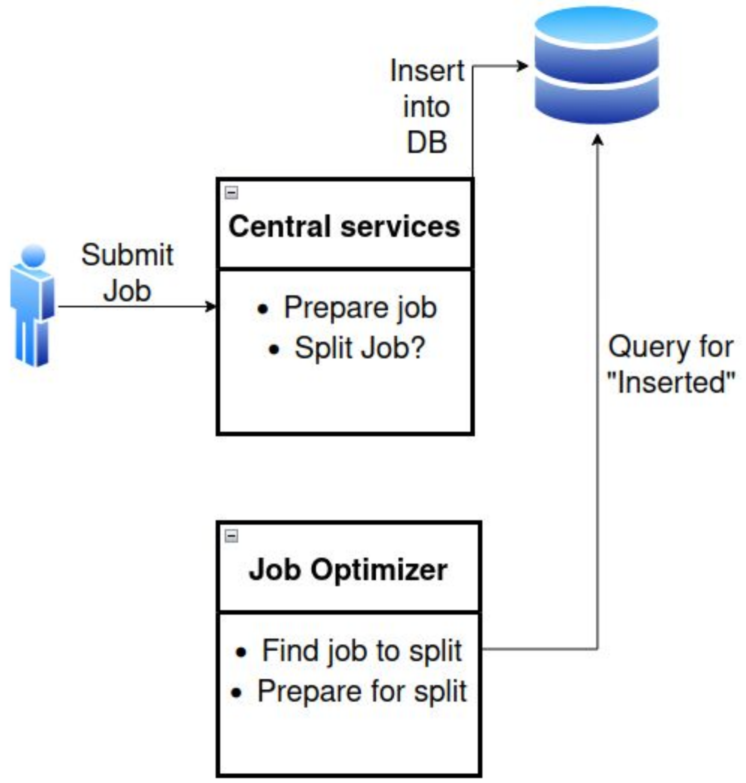
Workflow Job Optimizer

- Job id is needed, but update query returns no value
  - Want to do the operation in one query to alleviate db load
- Solution: Using user-defined variables
  - Can not get more than one job at a time
    - Returns latest value only
- Query for user defined variables without consulting table afterwards



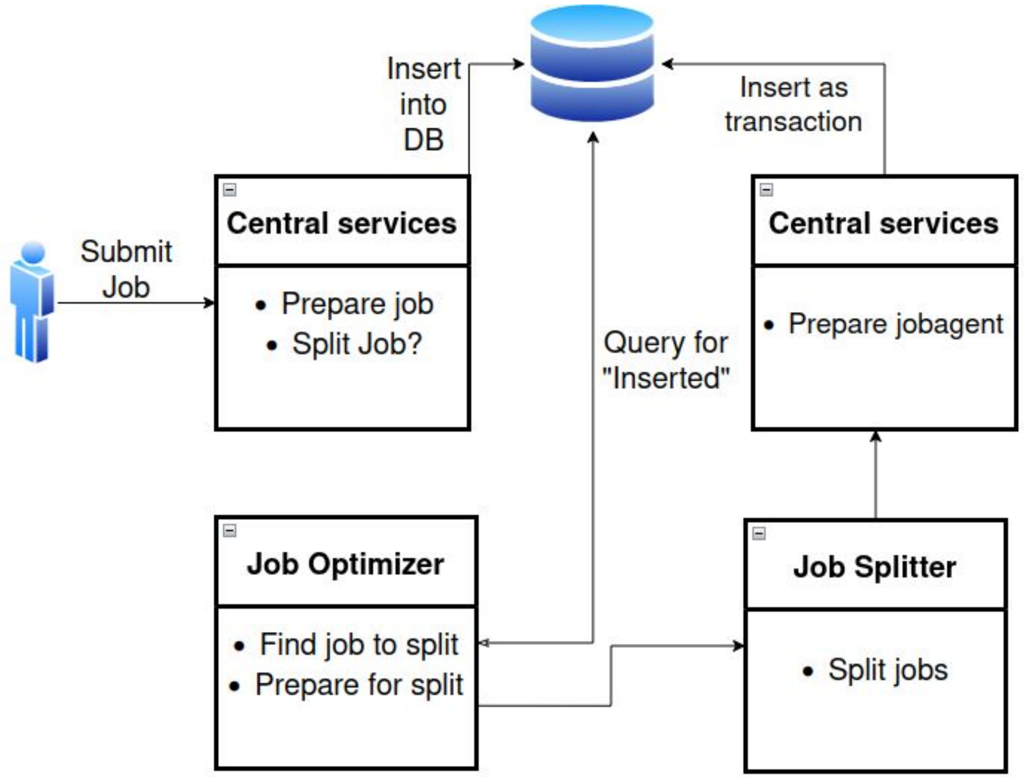
Workflow Job Optimizer

- If a job is stuck in the **Splitting** status for too long without finishing or producing errors, an action must occur
- Try splitting again
  - Retain time for this action
  - When inserting this time can be matched to last updated in database to ensure it is not added twice



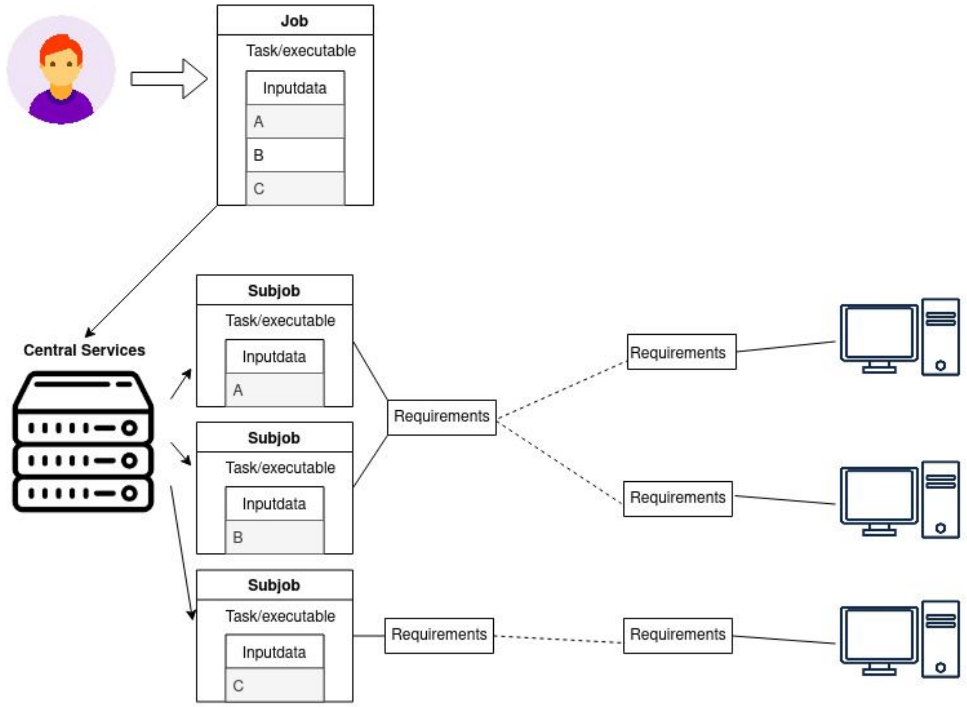
Workflow Job Splitter

- Prepare subjobs JDL before attempting to split
  - **#alien#**
    - Mention later
- Get input data (list or collection)
- Split into subjobs



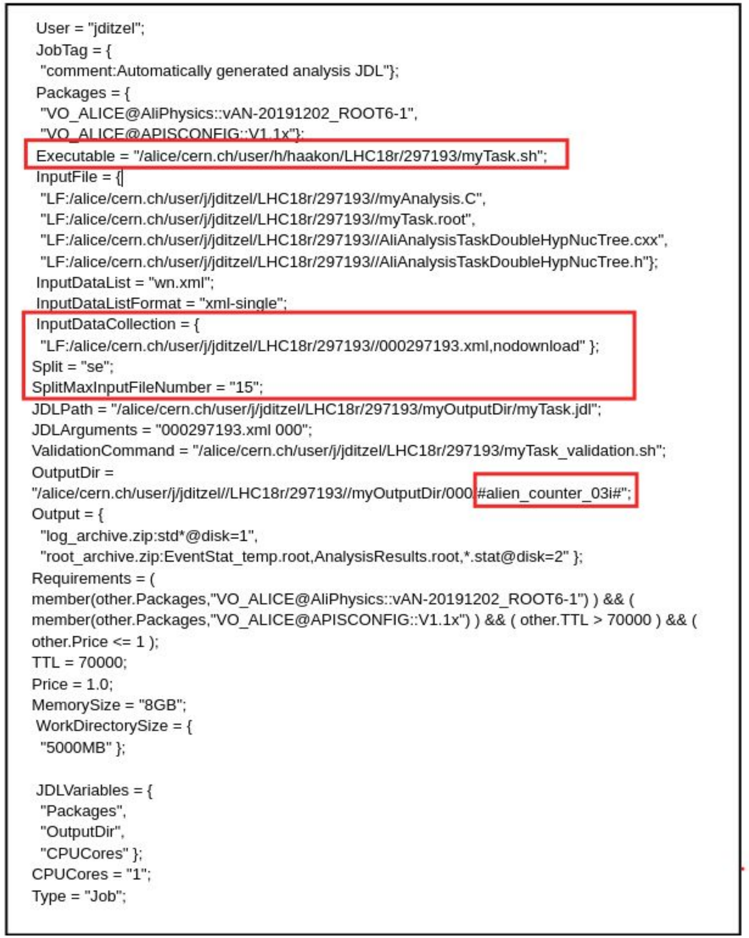
Splitting into subjobs

- Split is based on the inputdata
    - Subjobs will have different input data
    - Some exceptions



Subjob JDL

- Subjob JDL contains redundant information
  - Already exist in the masterjob
  - Takes up more resources than needed
    - Whole JDL stored as a varchar in the database
  - Solution: Describe subjobs as only the changes from masterjob
    - Apply changes to get full JDL
  - Not reflected in the database as of now, so the changes are applied before inserting into database



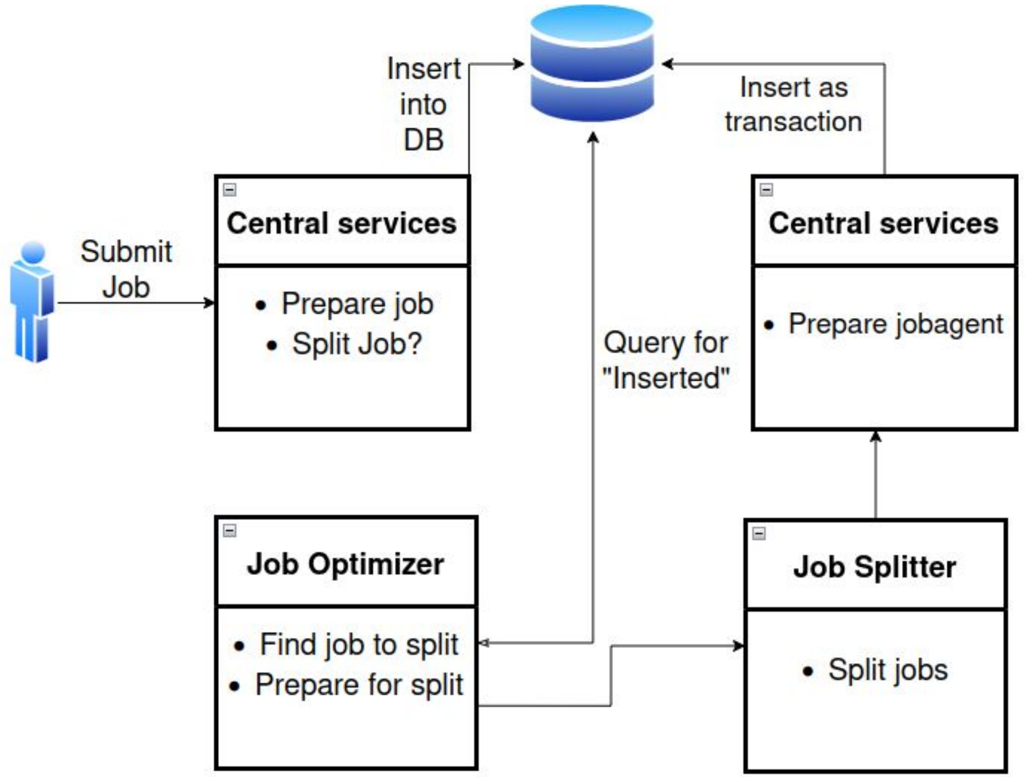
```
User = "jditzel";
JobTag = {
 "comment:Automatically generated analysis JDL"};
Packages = {
 "VO_ALICE@AliPhysics::vAN-20191202_ROOT6-1",
 "VO_ALICE@APISCONFIG::V1.1x"};
Executable = "/alice/cern.ch/user/h/haakon/LHC18r/297193/myTask.sh";
InputFile = {|
 "LF:/alice/cern.ch/user/j/jditzel/LHC18r/297193//myAnalysis.C",
 "LF:/alice/cern.ch/user/j/jditzel/LHC18r/297193//myTask.root",
 "LF:/alice/cern.ch/user/j/jditzel/LHC18r/297193//AliAnalysisTaskDoubleHypNucTree.cxx",
 "LF:/alice/cern.ch/user/j/jditzel/LHC18r/297193//AliAnalysisTaskDoubleHypNucTree.h"};
InputDataList = "wn.xml";
InputDataListFormat = "xml-single";
InputDataCollection = {
 "LF:/alice/cern.ch/user/j/jditzel/LHC18r/297193//000297193.xml,nodownload" };
Split = "se";
SplitMaxInputFileNumber = "15";
JDLPath = "/alice/cern.ch/user/j/jditzel/LHC18r/297193/myOutputDir/myTask.jdl";
JDLArguments = "000297193.xml 000";
ValidationCommand = "/alice/cern.ch/user/j/jditzel/LHC18r/297193/myTask_validation.sh";
OutputDir =
"/alice/cern.ch/user/j/jditzel//LHC18r/297193//myOutputDir/000#alien_counter_03i#";
Output = {
  "log_archive.zip:std*@disk=1",
  "root_archive.zip:EventStat_temp.root,AnalysisResults.root,*.stat@disk=2" };
Requirements = (
member(other.Packages,"VO_ALICE@AliPhysics::vAN-20191202_ROOT6-1") ) && (
member(other.Packages,"VO_ALICE@APISCONFIG::V1.1x") ) && ( other.TTL > 70000 ) && (
other.Price <= 1 );
TTL = 70000;
Price = 1.0;
MemorySize = "8GB";
 WorkDirectorySize = {
  "5000MB" };

 JDLVariables = {
  "Packages",
  "OutputDir",
  "CPUCores" };
CPUCores = "1";
Type = "Job";
```

Western Norway University of Applied Sciences

Workflow Job Splitter

- After JDL for subjobs are ready, prepare jobagents for subjobs and try to insert into Queue database
  - Use masterjob to populate relevant fields in the same insert query
  - Set status as **Waiting** to be ready for match with site
  - Status for masterjob is set to **Split**
- Contrary to old AliEn, this is done as one transaction.
  - All subjobs are inserted or none



JDL Split Fields

- JDL field **Split** defines if it is a splitjob, and how to split
- Split strategies:
  - Production
  - File, directory, parent directory
  - SE (Storage Element)
  - Custom
- Other Split arguments in the JDL includes:
  - **SplitMaxInputFileNumber** —> Set a limit for number of inputdata files per subjob
  - **SplitMinInputFileNumber** —> Set a minimum threshold for number of inputdata files per subjob, relevant to SE splitting
  - **SplitMaxInputFileSize** —> Set a limit to filesizes for inputdata files per subjob
  - Argument related to which SE to use?
    - Not implemented, can be effective if it is know that all input datafiles is found on one SE, or close by
  - **#alien#**

Production strategy

- Not a true split
- Same job duplicated a number of times
    - Monte carlo simulations
- Inputdata remains the same
- **Split = production:1-100**
    - Counter starts at lowest value

File, directory, parent directory strategy

- Inputdata files are split based on their LFN (Logical File Name)
  - Inputdata files under same directory might be grouped together
  - Splitting by **file** ensures that each subjobs have only one inputdata file
- **Split = file/directory/parentdirectory**

SE strategy

- Group inputdata based on locality
    - Starting with groups that share all SE’s
- Getting the SE’s for a large number of files can be time and resource consuming
    - Find LFN -> find GUID -> find PFN -> find real PFN -> find SE
    - Multiple database queries each step
    - To make it more efficient each steps take advantage of catalogue database being partitioned into different tables
        - Split the list of input into the corresponding tables, and do whole operations
            - Repeat for each step until you have real PFN

SE strategy

- After grouping inputdata by SE, some groups might include a 1000 inputdata files, while others only 1
    - Old AliEn did nothing to try and balance the load of the subjobs
    - JAliEn tries to group smaller groups together, based on one shared SE
        - SplitMinInputNumberFiles argument can set the threshold all groups must go voer
    - If there are none shared SE's, group together based on distance between SE
- **Split = SE**

Custom strategy

- The JDL for the subjobs must be defined in the JDL field **SplitDefinitions**
- Not been through real testing to ensure correctness as of now
- **Split = custom**
- **SplitDefinitions = {JDL for subjob1, JDL for subjob2}**

#alien# argument

- Replace **#alien#** in JDL with corresponding value
    - Based on what **#alien#** and subjob, counter most used
- In AliEn **#alien#** was limited to a few fields in JDL, Split Arguments, Outputdir, Output…
- JAliEn allows it to be defined anywhere in the JDL
    - Split Arguments is therefore redundant, but it still works as before
    - Doing a lot of matches to find and replace **#alien#** arguments in each subjob can be expensive
        - Solution: Do matching only on masterjob -> replace with lambda function
        - Run lambda function with correct input when building subjob JDL

#alien# argument

- **#alien_counter_03i#** —> 001, 002, 003…
    - #alien_counter# —> 1,2,3…
- **#aliendir#** —> /hremenes/input/inpudata = input
- **#alienfulldir#** —> /hremenes/input/inpudata = /hremenes/input/inputdata
- **#alienfilename/inputdata.root(oldvalue)/inputdata(newvalue)/#** —> /hremenes/input/inputdata.root = /hremenes/input/inputdata
- Use first, last or all in front to choose which inputdata file for the subjobs to use for the options above (Example: **#alienfirstdir#**)
    - “First” is default, uses first inputdata file, last the opposite
    - “All” uses all inputdata joined together with a “,” as delimiter

Western Norway University of Applied Sciences

Going forward

- Running smoothly on a local setup that interacts with database that contains production data for grid and catalogue
  - Still needs more testing to see where it breaks in an environment closer to production
- Next step is trying to run the job optimizer and job splitter in parallel and see the outcome
  - Fix eventual problems that arise

Thank you

Email: haakon.andre.reme-ness@cern.ch