Job submission using WP2-ERM (RLS) (1/3)

- the RB contacts the RLS in order to resolve logical file names and collect all the information about SEs containing at least one input data file

  SFN[ ] = edgReplicaManager::listReplicas(fileID)

- Where fileID is a LFN or a GUID specified in the InputData attribute of the JDL
- And a SFN is formed as follows

  SFN = sfn://sehost/path/filename

  E.g.: sfn://grid011g.pd.infn.it/datadir/subdir/track.dat

- From the SFNs gets the list of SEs containing replicas

Job submission using WP2-ERM (RLS) (2/3)

- Queries the IS to get the Ces:
  - On which the user is authorized
  - Are close to the SEs in the list that support at least one of the protocols in dataAccessProtocol
  - Having OutputSE as a close SE
- Using the information retrieved during the file name resolution, the RB classifies the CEs in groups according to the number of input files that are "near" to the CE.
- Take the first group (greatest num of files) and filter according to Requirements in the JDL (if no match is found go to next group)
- Choose the best one through the Rank in the JDL.

Job submission using WP2-ERM (RLS) (3/3)

- The RB writes the BrokerInfo file which contains all info the job needs in order to access the data, putting together the info returned by the WP2 API and those published into the IS

```
[
    CE = lxde01.pd.infn.it:2119/jobmanager-lsf-grid01;
    DataAccessProtocol = {"gridftp", "file"};
    PFNs = {"se1.pd.infn.it/data01/file762.dat"};
    LFN2PFN = [
        LFNs = {"file76.dat", "filex", "mndb"};
        PFNs = {
        {"se1.pd.infn.it/data00/file76.dat",  "se.cern.ch/cms/mn/file76.dat"},
        {"se1.pd.infn.it/data00/filex",  "se.in2p3.fr/data/00/filex"},
        {"se.cern.ch/cms/mn/cal1.DB"}
        };
          ];
    SEs = {"se1.pd.infn.it", "se.cern.ch", "se.in2p3.fr"};
    SEProtocols = {
        {"gridftp", "file"},  {"gridftp", "rfio"},  {"gridftp"}
          };
    SEPorts = {
    {4444, undefined}, #no port number for file protocol
    {4444, 5555},  {4433}
    };
    CloseSEs = {"se1.pd.infn.it", "se2.pd.infn.it"};
    SEMountPoint = {"/disk1", undefined}; # no local access to se2.pd.infn.it from this CE
    ]
```

Job submission using WP2-Opt. API (1/4)

- Usage of WP2 getAccessCost() API is triggered through the JDL
- Exploit the "classads functions" extension of the classads library: allows plugging external function to the library
- JDL: Rank = getAccesCost();
- For TB2.0 excludes other ranking expressions
- No other changes to the JDL from the users point of view

Job submission using WP2-Opt. API (2/4)

- Once a set of CE satisfying job requirements has been found, the following steps are performed:
  - The RB evaluate the Rank expression of the JDL, and hence calls:

AccessCost[ ] **edgReplicaManager::getAccessCost(LFNList[ ], CEList[ ], protocol)**

  - where LFNList is list of Logical File Names/GUIDs, CEList is list of Computing Elements and protocol is the protocol by which the files are accessed.

Job submission using WP2-Opt. API (3/4)

- The function returns for each input CE_n in CEList the following information:

CEn, {bestSFN$_{n1}$, bestSFN$_{n,2}$,..., bestSFN$_{n,m}$}, minTotalTime$_n$, sizeToBeReplicated

- where CE_n is the identifier of the n^{th} Computing Element, $bestSFN_{n,i}$ is the best replica for LFN_i with respect to CE_n, $minimumTotalCost_n$ is the minimum total access time of all best SFNs with respect to CE_n, and sizeToBeReplicated is the total size of best replicas that are remotely located.
- Discard a CE if it has no close SE with at least **sizeToBeReplicated** free space
- The **minTotalTime** value is used to rank the CEs: the best CE is the one having the shortest **minTotalTime**

Job submission using WP2-Opt. API (4/4)

- The list of best SFNs is used to write the BrokerInfo file to provide a hint for the job
- No file replication is triggered. RB will call the non-blocking version of getBestFile() when it will be available (not in TB2.0)
- For accessing a file from the WN the job must call the getBestFile() API.
  - The returned PFN is chosen independently from getAccessCost() as the environment may have changed significantly since the last function call.
- No file pinning is performed by the RB: pinning is done behind the scene by getBestFile().