FSR and DIRAC

Information required in DIRAC

- Currently all done by parsing the logfile
  - Too much dependent on message formats
- Analyse of application performance
  - In case of failure, identify where it happened
- Extract information for bookkeeping
  - Files processed successfully
    - Unreadable files are skipped
  - Events statistics
    - Events processed
    - Events written out in each output stream
- Additional information
  - Memory usage
  - Error statistics
  - Etc…

Meta example of how to pass information

- Processing application / version
  - { successful input files (GUID? LFN?) }
  - { failed input files (GUID? LFN?) }
  - Input events
  - Processed events
  - { (stream, events) }
  - [max. memory]
- Quite some overlap with ganga requirements
- Easier if processing information is also output in readable format (e.g. XML, pickled,…?)
  - Can also be extracted from output files
  - But then need to run a Gaudi application…
    - Advantage: checks the file is well formed
    - Disadvantage: cannot be used if application crashes

Q & A

- What type of object is going in the FSR (e.g. counters, ratios, averages)?
  - Counters and lists
- How are the objects accumulated (How are they identified, what operations are needed on them, and when (execute, finalize etc.))?
  - Counters: could be extracted from Gaudi services at finalise? File list should use incidents?
- In the job creating the objects, do they have to be accessed by algorithms other than the one accumulating them?
  - No
- Is there a need to propagate the objects to later processing steps (i.e. should they be copied to the output file if read from the input file)?
  - Probably yes
- Is there a need to combine objects from the FSR of several input files? If so, with what operations, and when?
  - No

- Is it possible to output FSR on a separate stream?
  - Can a file be updated on the fly?

Other usage for FSR

- User job finalisation
  - What to do if job is split?
  - Users want to see the same output as if the job was not split
- Output counters
    - ✰ No need to be included in output file, can be separate stream
  - Only cumulative counters
  - Allow merging of counters for split jobs
  - Is it useful for productions?
  - Is it useful for user jobs?
- How can one run the same finalisation code as in individual jobs on the merged counters?
  - Rerun the application without input files?

Conclusion

- FSR very useful to convey statistics on processing
- Probably useful to also output the same information in XML (DIRAC, ganga)
- How can it be used for merging finalisation?