HEP-WLs python parsers

Martina Javurkova

University of Massachusetts-Amherst

HEPiX Benchmarking Working Group

17/01/2020

Idea

- Have a common parser using ABC
- Advantages
  - Force all output JSON files to have a common structure which will be easy to read and interpret
  - Simplify a transition from shell parsers to python parsers for each experiment
  - Write all final scores in a scientific notation to overcome rounding issues
  - Possibility to have only one parser per experiment

JSON structure

- JSON files will have the following keys
  - wl-inputs: copies, threads_per_copy, events_per_thread
  - wl-scores: e.g. {“digi-reco”: 0.3490, “HITtoRDO”: 2.3456,…}
  - wl-stats: median, avg, min and max
  - wl-status: 1=failure or 0=success
  - wl-info: version, description, cvmfs_checksum, etc
  - wl-custom: each experiment can add whichever information

Example of methods (I)

- Methods which have **default** functionalities in a common parser (bmkParser.py)

  1) calculate_stats(self,appname,scores): score, median, avg, min, max

  2) save_outputs(self,appname,data,outfile): save to a summary JSON

  3) get_wl_inputs(self): get input parameters

  4) get_wl_scores(self,calculateVars): get the final score for a given WL

  5) get_wl_stats(self,calculateVars): get the statistics for a given WL

- Methods that should to be written by the **experiments** (atlasParser.py)

  1) check(self,inputdir)

    - Check if ALL input files are ready to be read i.e. can be opened and the job(s) finished successfully: True/False

Example of methods (II)

2) collect_values(self,inputdir)

- Collect all information from each input file and fill them to a dictionary with self-explained keys and the following structure
  - {“digi-reco”: {“realScore”:[…], “status”:[…]}, “HITtoRDO”:{…}}
- Mandatory keys
  - “status”
    - list of 0/1 containing status of each copy
  - “realScore” or “cpuScore” (or both)
    - list of scores of each process i.e. len(list)=NCOPIES*NTHREADS
    - If both are present, realScores are taken as default values

3) get_wl_custom(self,calculateVars): optional

Implementation

- Two possibilities

  1. Each experiment instantiates methods from ABC and writes python parser in a similar “fashion” as current ATLAS python parsers do

  2. Each experiment writes only two above mentioned methods and the rest will be taken care of by the common parsing script (can be called by the bmk-driver.sh)

➡Feedbacks from experiments are very welcome!