Experience running coffea-dask for ttbar analysis

Warnings:

```
distributed.worker - WARNING - Unmanaged memory use is high. This may indicate a memory leak or the memory may not be released to the OS; see h
ttps://distributed.dask.org/en/latest/worker.html#memtrim for more information. -- Unmanaged memory: 1.34 GiB -- Worker memory limit: 1.45 GiB
```

```
[#                                       ] | 3% Completed |  1min 22.1sTask exception was never retrieved
future: <Task finished name='Task-174' coro=<_wrap_awaitable() done, defined at /opt/conda/lib/python3.8/asyncio/tasks.py:688>
exception=AssertionError()>
Traceback (most recent call last):
  File "/opt/conda/lib/python3.8/asyncio/tasks.py", line 695, in _wrap_awaitable
    return (yield from awaitable.__await__())
  File "/opt/conda/lib/python3.8/site-packages/distributed/deploy/spec.py", line 60, in _
    assert self.status == Status.running
AssertionError
```

```
  warnings.warn(
DCSchedd::spoolJobFiles:7002:File transfer failed for target job 35705899.0: TOOL at 131.225.189.90 failed to send file(s) to
<131.225.188.57:9618>; SCHEDD at 131.225.188.57 failed to receive file /storage/local/data1/condor/spool/5899/0
/cluster35705899.proc0.subproc0.tmp/JEC/Summer19UL18_V5_MC/Summer19UL18_V5_MC_UncertaintySources_AK8PFchs.txt
```

Error: Fatal, Recurring but disappearing

```
/opt/conda/lib/python3.8/concurrent/futures/_base.py in result(self, timeout)
    435                     raise CancelledError()
    436                 elif self._state == FINISHED:
--> 437                     return self.__get_result()
    438
    439                 self._condition.wait(timeout)

/opt/conda/lib/python3.8/concurrent/futures/_base.py in __get_result(self)
    387         if self._exception:
    388             try:
--> 389                 raise self._exception
    390             finally:
    391                 # Break a reference cycle with the exception in self._exception

BrokenProcessPool: A process in the process pool was terminated abruptly while the future was running or pending.
```

Investigating using ‘python memory profile’

Fatal Error: (also happens with simple_exampe.py)

```
Traceback (most recent call last):
  File "TTbarDileptonicAnalysis.py", line 53, in <module>
       hists, metrics = processor.run_uproot_job(
  File "/opt/conda/lib/python3.8/site-packages/coffea/processor/__init__.py", line 104, in _run_x_job
       return run(
  File "/opt/conda/lib/python3.8/site-packages/coffea/processor/executor.py", line 1337, in __call__
       wrapped_out = executor(chunks, closure, None)
  File "/opt/conda/lib/python3.8/site-packages/coffea/processor/executor.py", line 725, in __call__
       else _decompress(work.result())
  File "/opt/conda/lib/python3.8/site-packages/distributed/client.py", line 238, in result
       raise exc.with_traceback(tb)
distributed.scheduler.KilledWorker: ('TTbarDileptonProcessor-79d46cb336bafded7fd9fcbea5a1c3d5', <WorkerState
'tcp://131.225.188.14:10000', name: LPCCondorCluster-0, status: closed, memory: 0, processing: 38>)
>>>
Last-ditch attempt to close HTCondor job 35705911 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705910 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705908 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705907 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705906 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705905 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705903 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705902 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705900 in finalizer! You should confirm the job exits!
Last-ditch attempt to close HTCondor job 35705898 in finalizer! You should confirm the job exits!
Task exception was never retrieved
future: <Task finished name='Task-3826' coro=<LPCCondorJob.close() done, defined at
/srv/.env/lib/python3.8/site-packages/lpcjobqueue/cluster.py:134> exception=RuntimeError('cannot schedule new futures after
shutdown')>
Traceback (most recent call last):
  File "/srv/.env/lib/python3.8/site-packages/lpcjobqueue/cluster.py", line 158, in close
       if await asyncio.get_event_loop().run_in_executor(None, check_gone):
  File "/opt/conda/lib/python3.8/asyncio/base_events.py", line 783, in run_in_executor
       executor.submit(func, *args), loop=self)
  File "/opt/conda/lib/python3.8/concurrent/futures/thread.py", line 179, in submit
       raise RuntimeError('cannot schedule new futures after shutdown')
RuntimeError: cannot schedule new futures after shutdown
```

Suggestion, but does not work.

```
del client
cluster.close()
```

Questions:

- I have noticed sometimes that process end up running on dask (or locally on futures) in the background even after closing the jupyter notebook, how do we monitor these processes? And check if they are running.
- Certain datasets take ~8 hours or more with (10 -> 4) workers to finish with the current processor. Is there a way to speed this up?
- "**skipbadfiles**" parameter for reading Data files. What are the options to resubmit failed jobs?

```
/opt/conda/lib/python3.8/site-packages/coffea/processor/executor.py:965: UserWarning: file not found ([ERROR] Serve
r responded with an error: [3011] No servers are available to read the file.
)

    'root://cmsxrootd-site.fnal.gov//store/mc/RunIISummer20UL18NanoAODv9/DYJetsToLL_M-10to50_TuneCP5_13TeV-madgraph
MLM-pythia8/NANOAODSIM/106X_upgrade2018_realistic_v16_L1v1-v1/280000/9B4572B9-1ED0-C14C-9991-8C63ED3B0D7D.root'

Files may be specified as:
   * str/bytes: relative or absolute filesystem path or URL, without any colons
         other than Windows drive letter or URL schema.
         Examples: "rel/file.root", "C:\abs\file.root", "http://where/what.root"
   * str/bytes: same with an object-within-ROOT path, separated by a colon.
         Example: "rel/file.root:tdirectory/ttree"
   * pathlib.Path: always interpreted as a filesystem path or URL only (no
         object-within-ROOT path), regardless of whether there are any colons.
         Examples: Path("rel:/file.root"), Path("/abs/path:stuff.root")

Functions that accept many files (uproot.iterate, etc.) also allow:
   * glob syntax in str/bytes and pathlib.Path.
         Examples: Path("rel/*.root"), "/abs/*.root:tdirectory/ttree"
   * dict: keys are filesystem paths, values are objects-within-ROOT paths.
         Example: {"/data_v1/*.root": "ttree_v1", "/data_v2/*.root": "ttree_v2"}
   * already-open TTree objects.
   * iterables of the above.

  warnings.warn(str(e))
```