Using the Work Queue Executor

coffea users meeting August 2021

Ben Tovar     btovar@nd.edu
Work Queue Executor Overview

coffeas as wq manager (local)

item descriptions
process functions
or accumulator function with results to accumulate
python environment (once per worker)

worker (remote)

root files

function results

xrootd
Work Queue Executor User Responsibilities

- Setup coffea as Work Queue manager: **executor args**
- Construct a tarball with the python environment: **conda recipe**
- Launch desired number of workers in batch system: **manually or with wq factory**

- Optional, but highly recommended:
  - Activate automatic resource management: **wq figures out cores, mem, disk per function.**
  - Activate all log types.
Minimal Work Queue Executor arguments

```python
from processor import run_uproot_job, work_queue_executor
exec_args = {
    # give the manager a name so workers can automatically find it:
    'master_name': '{}-wq-coffea'.format(os.environ['USER']),
    # find a port to run work queue in this range (above 1024):
    'port': [9123, 9130],
    # if not given, assume environment already setup at execution site
    'environment_file': 'my-coffea-env.tar.gz',
    # only if different from /tmp/x509up_u...
    # 'x509_proxy': 'myproxy.pem'
    # almost all coffea functions use one core
    'cores': 1
}
out = run_uproot_job(..., executor=work_queue_executor, executor_args=exec_args)
```
Constructing the Environment (if not setup at execution site)

# First install miniconda if you don't yet have conda or anaconda.
$ curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh > conda-install.sh
$ bash conda-install.sh
$ ...follow conda instructions...

# Create the environment:
$ unset PYTHONPATH
$ conda create --name my-coffea-env python=3.??

# Install needed modules
$ conda activate my-coffea-env
$ conda install -c conda-forge coffea xrootd ndcctools dill conda-pack
$ ...pip (not editable) or conda install other packages you need...

# Create the portable environment (everytime app or coffea is updated)
$ conda activate my-coffea-env
$ conda-pack --output my-coffea-env.tar.gz
Running with one local worker

$ conda activate my-coffea-env
$ voms-proxy-init2  # only if you need certs to access data
$ python my-coffea-env-application

Listening for work queue workers on port 9124...
warning: this work queue manager is visible to the public.
warning: you should set a password with the --password option.
submitted preprocessing task 1
submitted preprocessing task 2
submitted preprocessing task 3
...

# In another terminal, four tasks per worker
$ conda activate my-coffea-env
$ work_queue_worker --cores 4 --memory 8000 --disk 4000 -M ${USER}-wq-coffea

work_queue_worker: creating workspace /tmp/worker-196886-2912809
work_queue_worker: using 4 cores, 8000 MB memory, 4000 MB disk, 0 gpus
connected to manager 10.32.77.18:9124 via local address 10.32.77.18:39084
Running workers on batch system, manually

# a hundred workers in a campus cluster, 400 concurrent tasks
$ conda activate  my-coffea-env
$ condor_submit_workers --cores 4 --memory 8000 --disk 4000 -M ${USER}-wq-coffea 100

# one worker on stampede2, 68 concurrent tasks
# explicitly specify coffea wq manager host xxx.xxx... and port yyyy
# specify slurm parameters, normal queue, 60 minutes
$ conda activate  my-coffea-env
$ slurm_submit_workers -s $SCRATCH/workers -p '-p normal -t 60' xxx.xxx... yyyy 1
import work_queue

exec_args = {
    'master_name': '{}-wq-coffea'.format(os.environ['USER']),
    ...
}

# available: local, condor, sge, slurm
workers = work_queue.Factory("condor", exec_args["master_name"])
workers.cores = 4
workers.memory = 16000
workers.disk = 20000
workers.max_workers = 20
workers.min_workers = 1

with workers:
    output = processor.run_uproot_job(
        ...
        executor=processor.work_queue_executor,
        executor_args=exec_args,
        ...
    )
Using the Work Queue factory (external)

# wq factory creates workers as needed.
$ conda activate my-coffea-env
$ work_queue_factory -Tcondor -C my-conf.json

my-conf.json: (re-read when changed, useful to manipulate max and min workers.)

```json
{
    "manager-name": "btovar-wq-coffea",
    "max-workers": 200,
    "min-workers": 1,
    "workers-per-cycle": 20,
    "cores": 4,
    "memory": 16000,
    "disk": 20000,
    "timeout": 900,
}
```
Automatic resource allocations

```python
evac傳統 = {
    'master_name': '{}-wq-coffea'.format(os.environ['USER']),
    'port': [9123, 9130],
    'environment_file': "my-coffea-env.tar.gz",

    # no task can use more than these maximum values:
    'cores': 2,
    'memory': 8000,
    'disk': 8000,

    # adjust size of resources allocated to tasks as they are measured
    # use maximum resources seen, retry on maximum values if exhausted.
    'resource_monitor': True,
    'resources_mode': 'auto',
}
Example of resource allocation

TASK 1333 RUNNING 10.32.86.105:58248  FIRST_RESOURCES {"memory":750,"MB"},"disk":500,"MB"},"gpus":0,"gpus"},"cores":1,"cores"}...

TASK 1333 RETRIEVED RESOURCE_EXHAUSTION  143 {"memory":750,"MB"} {...

TASK 1333 RUNNING 10.32.86.105:58248  MAX_RESOURCES {"memory":8000,"MB"},"disk":8000,"MB"},"gpus":0,"gpus"},"cores":2,"cores"}...

processing task id 1333 item 8026473b with 127000 events completed on d12chas555.crc.nd.edu. return code 0 allocated cores: 2.0, memory: 8000.0 MB, disk: 8000.0 MB, gpus: 0.0 measured cores: 0.368, memory: 1191.0 MB, disk 305.0 MB, gpus: 0.0, runtime 329.988136

processing task id 1756 item fa13269a with 117845 events completed on d12chas580.crc.nd.edu. return code 0 allocated cores: 1.0, memory: 4250.0 MB, disk: 500.0 MB, gpus: 0.0 measured cores: 0.353, memory: 4000.0 MB, disk 310.0 MB, gpus: 0.0, runtime 310.851779
Enable debug logs

exec_args = {
    ...
    # print messages when tasks are submitted, and as they return, their
    # resource allocation and usage.
    'verbose': True,
    # detailed debug messages
    'debug_log': 'debug.log',
    # lifetime of tasks, workers, and resource allocations
    'transactions_log': 'tr.log',
    # time series of manager statistics, plot with work_queue_graph_log
    'stats_log': 'stats.log',
}
verbose': True

code 0  
allocated cores: 1.0, memory: 2750.0 MB, disk: 500.0 MB, gpus: 0.0  
measured cores: 0.379, memory: 1593.0 MB, disk 300.0 MB, gpus: 0.0, runtime 200.515746  
submitted processing task id 2095 item 974bfe68, with 123500 events  
processing task id 1396 item dc66182a with 127750 events completed on d12chas583.crc.nd.edu. return code 0  
allocated cores: 1.0, memory: 2500.0 MB, disk: 500.0 MB, gpus: 0.0  
measured cores: 0.4, memory: 1357.0 MB, disk 291.0 MB, gpus: 0.0, runtime 239.530528  
submitted processing task id 2096 item 0894bedf, with 117000 events  
processing task id 1549 item 19e56faef with 116357 events completed on d12chas547.crc.nd.edu. return code 0  
allocated cores: 1.0, memory: 2750.0 MB, disk: 500.0 MB, gpus: 0.0  
measured cores: 0.363, memory: 1206.0 MB, disk 300.0 MB, gpus: 0.0, runtime 215.822369  
submitted processing task id 2097 item f303c8d5, with 117000 events  
processing task id 1601 item f91400b4 with 111359 events completed on d12chas575.crc.nd.edu. return code 0  
allocated cores: 1.0, memory: 2750.0 MB, disk: 500.0 MB, gpus: 0.0  
measured cores: 0.368, memory: 1080.0 MB, disk 300.0 MB, gpus: 0.0, runtime 209.20616  
submitted processing task id 2098 item c1da1e8d, with 127500 events  
processing task id 1406 item fd51ce29 with 123000 events completed on d12chas560.crc.nd.edu. return code 0  
allocated cores: 1.0, memory: 2500.0 MB, disk: 500.0 MB, gpus: 0.0  
measured cores: 0.384, memory: 2431.0 MB, disk 291.0 MB, gpus: 0.0, runtime 242.325099  
submitted processing task id 2099 item 7f66cfa1, with 127500 events

Submitted 53% | 126436654/237149394 [06:30<19:32, 94461.19event/s]  
Processing 13% | 31814581/237149394 [06:30<35:37, 96052.40event/s]  
Accumulated 13% | 10/75 [06:30<23:50, 22.01s/tasks]
Plot stats.log

$ conda activate my-coffea-env
$ work_queue_graph_log -Tpng stats.log
Other exec args: 'compression': 9 (default)

compression: None  compression: 0 (min)  compression: 9 (mid)  compression: 16 (max)

Manager Time

Tasks instantaneous counts

waiting on workers
running
with results

manager data transfer

GB

GB

GB
Size of accumulation tasks

# these are the default values

exec_args = {
    ...
    # use mid-range compression for chunks results. 9 is the default for work
    # queue in coffea. Valid values are 0 (minimum compression, less memory
    # usage) to 16 (maximum compression, more memory usage).
    'compression': 9,

    # control the size of accumulation tasks. Results are
    # accumulated in groups of size chunks_per_accum, keeping at
    # most chunks_per_accum at the same time in memory per task.
    'chunks_per_accum': 10,
    'chunks_accum_in_mem': 2,
}
In development

- Automatic chunksize with target memory usage per task.
  - Currently can be used with `run_uproot_job(..., dynamic_chunksize=True)`
  - chunksize adapted so that tasks run for about 1 minute by default

- Automatic construction of python environment when changes are detected.
  - Currently topcoffea (a coffea application) uses an automatic construction.
  - Picks up local changes in pip editable packages, useful for development.
  - We are working on generalizing it for coffea.

WARNING: Found unstaged changes in '.../src/topcoffea'
INFO: Looking for base environment env/base_env_a97aaafa0.tar.gz...
INFO: Creating environment /tmp/tmpn53zo2eo
INFO: Installing python=3.8.10, conda into /tmp/tmpn53zo2eo via conda
INFO: Installing conda-pack, dill, xrootd into /tmp/tmpn53zo2eo via conda
INFO: Installing reqs of .../src/coffea into /tmp/tmpn53zo2eo via pip
INFO: Installing reqs of .../src/topcoffea into /tmp/tmpn53zo2eo via pip...
executor_args = {
    'master_name': '{}-workqueue-coffea'.format(os.environ['USER']),
    'port': [9123, 9130],
    'debug_log': 'debug.log',
    'transactions_log': 'tr.log',
    'stats_log': 'stats.log',
    'environment_file': topeftenv.get_environment(),
    'extra_input_files': ['topeft.py'],
    'schema': NanoAODSchema,
    'skipbadfiles': False,
    'resource_monitor': True,
    'resources_mode': 'auto',
    'cores': 1,
    'disk': 8000,  # MB
    'memory': 8000,  # MB
    'fast_terminate_workers': 5,  # terminate workers which seem slower than the rest (avg * 5)
    'chunks_per_accum': 25,
    'verbose': True,
}
Quick comparison with parsl and dask executors

wq:

Easier to develop when no shared filesystem is available, opportunistic resources.

Tarball with python environment is important (conda-pack).

Automatic resource management, and dynamic chunksize.

Dynamic accumulation as results are computed.

parsl:

wq is also available as a parsl provider.

'workers' declared in code to configure the provider.

Parsl executor assumes you already setup the environment at the execution site.

dask:

As wq, workers can be declared in python, or outside the coffea code.

Dask executor assumes you already setup the environment at the execution site.
More information

btovar@nd.edu

work queue documentation:

https://cctools.readthedocs.io/en/latest/work_queue