Software for HAL

Slurm Wrapper Suite

Introduction

The HAL Slurm Wrapper Suite is designed to help users use the HAL system easily and efficiently. The current version is "swsuite-v1.0", which includes

```
srun ? swrun : request resources to run interactive jobs.
sbatch ? swbatch : request resource to submit a batch script to slurm.
squeue ? swqueue : check current running jobs and computational resource status.
```

Goals

- Minimize the required input options.
- Consistent with the original "slurm" run-script format.
- Submits job to suitable partition based on the number of CPUs and/or GPUs needed.

Usage

Interactive Mode

```
swrun [-h] -p PARTITION [-c CPU_PER_GPU] [-t TIME] [-s SINGULARITY][-r RESERVATION] [-v]
```

- partition (required) : cpu_mini, cpun1, cpun2, cpun4, cpun8, cpun16. gpux1, gpux2, gpux3, gpux4, gpux8, gpux12, gpux16.
- [cpu_per_gpu] (optional) : 16 cpus (default), range from 16 cpus to 40 cpus.
- [time] (optional) : 24 hours (default), range from 1 hour to 72 hours (walltime).
- [singularity] (optional): specify a singularity container(name-only) to use from the $HAL_CONTAINER_REGISTRY
- [reservation] (optional): specify a reservation name, if any.
- [version] (optional): Display Slurm wrapper suite and Slurm versions.

Example

To request a full node: 4 gpus, 160 cpus (40*4 = 160 cpus) , 72 hours

```
swrun -p gpux4 -c 40 -t 72
```

or using a container image (dummy.sif) on a cpu only node with default of 24 hours

```
swrun -p cpun1 -s dummy
```

Note: In the second case we are using a singularity container image. To run a custom container, instead of using the default location of the container registry, you can set it to your own by first exporting the environment variable

```
export HAL_CONTAINER_REGISTRY="/path/to/custom/registry"
```

Script Mode

```
swbatch [-h] RUN_SCRIPT [-v]
```

Same as original slurm batch.

- RUN_SCRIPT (required) : Specify a batch script as input.

Within the run_script:

- partition (required) : cpu_mini, cpun1, cpun2, cpun4, cpun8, cpun16, gpux1, gpux2, gpux3, gpux4, gpux8, gpux12, gpux16.
- job_name (optional) : job name.
- output_file (optional) : output file name.
- error_file (optional) : error file name.
- cpu_per_gpu (optional) : 16 cpus (default), range from 16 cpus to 40 cpus.
- time (optional) : 24 hours (default), range from 1 hour to 72 hours (walltime).
- singularity (optional) : Specify a singularity image to use. The container image is searched for from the container registry directory environment variable in the swconf.yaml configuration.

Example:

Consider demo.swb, which is a batch script such as

```
#!/bin/bash
#SBATCH --partition=gpux1

srun hostname
```

or using a container image with a time of 42 hours

```
#!/bin/bash
#SBATCH --job_name="demo"
#SBATCH --output="demo.%j.%N.out"
#SBATCH --error="demo.%j.%N.err"
#SBATCH --partition=gpux1
#SBATCH --time=42
#SBATCH --singularity=dummy

srun hostname
```

You can run the script as below but remember to export the container registry variable if you are using some custom singularity images.

```
swbatch demo.swb
```

Monitoring Mode

```
swqueue
```

Same as original slurm squeue, which show both running and queueing jobs, the swqueue shows running jobs and computational resource status.

Tar2h5

Convert Tape ARchives to HDF5 files

- **archive_checker**: check how many files can be extracted from the input tar file.
- **archive_checker_64k**: check if any files within input tar files larger than 64 KB.
- **h5compactor**: convert input tar file into hdf5 file, all files within tar file should smaller than 64KB, using small files name as dataset names.
- **h5compactor-sha1**: convert input tar file into hdf5 file, all files within tar file should smaller than 64KB, using small files sha1 values as dataset names.
- **h5shredder**: convert input tar file into hdf5 file, no file size limitation, concatenate data and offsets into 4 seperate arrays for better randomized access.

Install dependent packages (on Ubuntu-20.04-LTS)

- hdf5

```
sudo apt install libhdf5-103 libhdf5-dev libhdf5-openmpi-103 libhdf5-openmpi-dev
```

- libarchive

```
sudo apt install libarchive13 libarchive-dev
```

- cmake

```
sudo apt install cmake
```

- openmpi

```
sudo apt install libopenmpi3 libopenmpi-dev openmpi-bin
```

- libssl

```
sudo apt install libssl1.1 libssl-dev
```

Install tar2h5 software

```
github clone https://github.com/HDFGroup/tar2h5.git
cd tar2h5
cmake .
make
```

Uninstall tar2h5 software

```
make clean
Run CTest
ctest
```

Run tar2h5 software

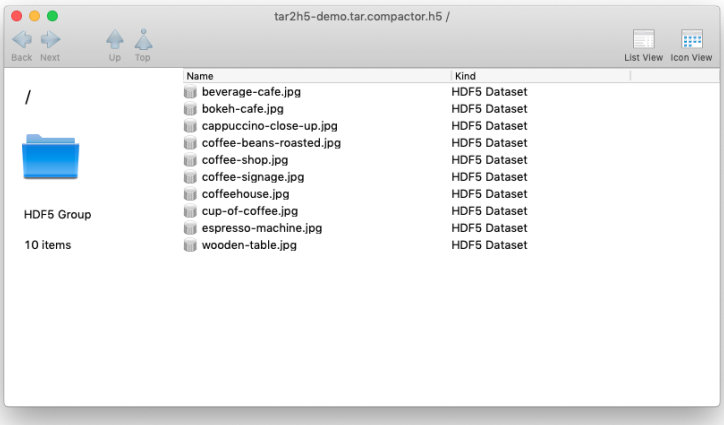
```
./bin/archive_checker ./demo/tar2h5-demo.tar
./bin/archive_checker_64k ./demo/tar2h5-demo.tar
./bin/h5compactor ./demo/tar2h5-demo.tar
./bin/h5compactor-sha1 ./demo/tar2h5-demo.tar
./bin/h5shredder ./demo/tar2h5-demo.tar
```

Visualization with HDFCompass

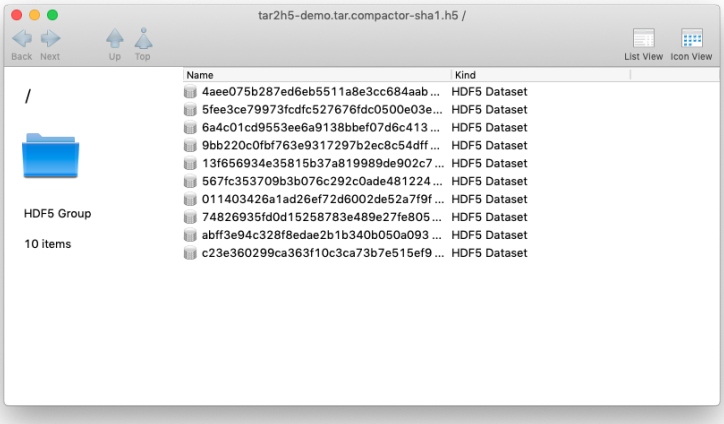
https://support.hdfgroup.org/projects/compass/

Output File Format

- compactor output sample



- compactor-sha1 output sample



- shredder output sample