DESI Fiber Assignment for Mocks

2019-01-29

Ted Kisner • LBNL • tskisner@lbl.gov

Recent Fiberassign Developments

- Hybrid C++ / Python package: more low-level control from Python
- C++ used for calculation, OpenMP threaded where possible
- All I/O is done from Python
- Program ”flow” happens at the Python layer
- C++ classes ”look like” Python classes (pybind11)
- Many timing / debugging options.

Example Timings

Running the DR7.1 data (65M total targets, 10K tiles) on one cori node and using about half the available RAM:

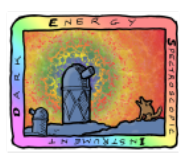
- Read hardware / tile info (2s)
- Read target data (60s)
- Compute targets available to every tile / fiber (54s)
- Compute tile and fibers available to every target (190s)
- Assign unused fibers to science targets (443s)
- Load balance science targets across future tiles (250s)
- Assign unused fibers to standards, to some limit (65s)
- Forcibly assign standards (bump science) to some limit (105s)
- Assign unused fibers to sky, to some limit (80s)
- Forcibly assign sky (bump science) to some limit (444s)
- Assign remaining unused fibers to science targets (52s)
- Assign remaining unused fibers to sky targets (62s)

PIP Weights

Pairwise Inverse Probability (PIP) weights- see discussion in papers here and here. For some number of realizations sufficient to sample the selection probability, each PIP weight can be computed from a bit array for the 2 targets:

$$W_{ij} = \frac{N_{\text{realization}}}{\text{popcount}\left[w_i \;\&\; w_j\right]}$$

The bit arrays have one element per realization which are set to one if the target was assigned and zero otherwise. For each realization, the tiling is dithered and some targets have their priority class upweighted.



PIP Weights (Implementation)

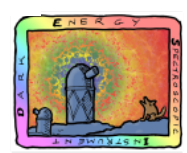
- Read data common to all realizations (hardware specs, targets)
- For each realization, dither tiling and run
- Accumulate bitarray weight vectors for every target
- Run one or a few realizations per compute node
- Save the per-target bitarrays to disk

For a ”DR7 sized” dataset, previous tests indicate about 1/2 node-hour per realization. So 1000 realizations could run on (say) 250 nodes for 2 hours.

PIP Weights (Example)

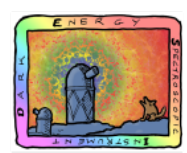
We can write a custom script for computing the weights using mpi4py to run fiber assignment with one process per node.

```
#!/usr/bin/env python3

from mpi4py import MPI
import numpy as np
from bitarray import bitarray
from fiberassign.hardware import load_hardware
from fiberassign.tiles import load_tiles
from fiberassign.targets import (TARGET_TYPE_SCIENCE,
                                 TARGET_TYPE_SKY, TARGET_TYPE_STANDARD,
                                 Targets, TargetsAvailable,
                                 TargetTree, FibersAvailable,
                                 load_target_file)
from fiberassign.assign import Assignment

# Read hardware properties
hw = load_hardware()

# Read the nominal footprint
nominal_tiles = load_tiles(hw)

# Target files.  Get these from command line arguments, etc.
target_files = [
    "mtl.fits",
    "sky.fits"
]
```



PIP Weights (Example Continued)

```
# Realizations to do- read these from a file or some other source.
realizations = np.arange(1000, dtype=np.int32)

# How many MPI processes do we have and what is our rank?
mpi_procs = MPI.COMM_WORLD.size
mpi_rank = MPI.COMM_WORLD.rank

# Load Target data.  There may be some contention from all
# the MPI processes reading at once, but this is only done
# one time at the start of the job.
tgs = Targets()
for tgfile in target_files:
    load_target_file(tgs, tgfile)

# Create a hierarchical triangle mesh lookup of the targets positions
tree = TargetTree(tgs)

# What iterations should this MPI rank compute?
my_realizations = np.array_split(realizations, mpi_procs)[mpi_rank]

# Initialize the bitarray per target for our local realizations...

# Every MPI rank works on its realizations.  We run this job with one MPI rank
# per node and then use OpenMP and multiprocessing within a node.
for realization in my_realizations:
    # Based on this realization index, set up numpy random seeds, etc.
    # If we are dithering our tile RA/DEC compute that here.
    tiles = dither_tiles(nominal_tiles, seed)
```



PIP Weights (Example Continued)

```
        # Compute the targets available to each fiber for each tile.
        tgsavail = TargetsAvailable(tgs, tiles, tree)

        # Compute the fibers on all tiles available for each target
        favail = FibersAvailable(tgsavail)

        # Create assignment object
        asgn = Assignment(tgs, tgsavail, favail)

        # First-pass assignment of science targets
        asgn.assign_unused(TARGET_TYPE_SCIENCE)

        # Redistribute science targets across available petals
        asgn.redistribute_science()

        # Assign standards, 10 per petal
        asgn.assign_unused(TARGET_TYPE_STANDARD, 10)
        asgn.assign_force(TARGET_TYPE_STANDARD, 10)

        # Assign sky, up to 40 per petal
        asgn.assign_unused(TARGET_TYPE_SKY, 40)
        asgn.assign_force(TARGET_TYPE_SKY, 40)

        # If there are any unassigned fibers, try to place them somewhere.
        asgn.assign_unused(TARGET_TYPE_SCIENCE)
        asgn.assign_unused(TARGET_TYPE_SKY)

        # Loop over assigned tiles and update our bit array for this realization

    # Gather bitarrays from all processes to rank zero and write them out.
```