Description of Post-Processing Tasks to Generate LCMAP Visual Change Products

Brian Davis, SGT, Inc.
11/13/2017

Step 1: Download PyCCD change results and values as .json formatted files.

Python Module Environment:

pip freeze:

cassandra-driver==3.10
GDAL==2.2.1
numpy==1.13.1
PyYAML==3.12
requests==2.14.2
scikit-learn==0.19.0
scipy==0.19.1
six==1.11.0

Scripts Used:

- manage_pyccd_results.py
- extent_from_hv.py
- (or, iPython notebook "PyCCD json.ipynb)

Summary:

PyCCD results are written to and stored by the Cassandra cluster. They must be retrieved and stored as .json disk files to enable creation of change map products, until new automation tools are in place (FireBird).

Check for Complete Chips**:**

python manage_pyccd_results.py <site> <H> <V> <version> <action>

site:	the test site name in which the tile-stack resides
H:	the Horizontal grid index number of the tile-stack
V:	the Vertical grid index number of the tile-stack
version:	the PyCCD version for which to poll status of results
action:	valid actions are 'count', 'retrieve', 'request'
	count reports completed, incomplete, and missing chips
	retrieve downloads list of complete chips results to .json files
	request submits requests for the missing chips to be processed, and are sent to the rabbitMQ queue

All 2,500 chips in a tile-stack are required for change map generation. As chips are completed on the mesos cluster, as reported by RabbitMQ, they can be polled to distinguish status when multiple tile-stacks are in process. Elapsed time for counting completed chips can take several to fifteen minutes using 2 to 4 threads, depending on system utilization by other tasks. Example syntax:

```
python manage_pyccd_results.py il 20 7 lcmap-pyccd:2017.08.18 count
```

Estimated Time (h:mm:ss):

0:00:15

Retrieve Complete Chips**:**

After all 2,500 chips in a tile-stack are completed, they can be downloaded for Q/A and subsequent change map generation. Elapsed time varies with number of threads defined. Example syntax:

```
python manage_pyccd_results.py il 20 7 lcmap-pyccd:2017.08.18 retrieve
```

A much simpler, limited use capability for downloading .json files can be accomplished by editing and running the following script:

```
python firebird_json_results.py
```

Estimated Time (h:mm:ss):

1:45:00

Step 2: Verify the results in each of the 2,500 chips in a tile-stack:

Scripts Used:

- count_json_results.py

Summary:

Occasionally, there have been issues with the Cassandra and/or Mesos clusters. To ensure PyCCD has processed all chips successfully before using those results for subsequent product generation, some quality checks are performed. All 10,000 X-Y locations are verified to have a successful status, which also will catch any locations which are completely missing from the .json file. This is possible if there was an unreported problem with the SEE during processing. Any X-Y locations with a “result_ok”key with a value of anything but true, along with X-Y locations which are missing, are written to stdout, along with the values of X and Y, to identify problems to resolve. stdout and/or stderr can be redirected to an output text file for saving messages. This file is typically saved with the .json files for future reference.

```
python count_json_results.py <path-to-dir-of-json-files>
```

Example Syntax

python count_json_results.py /data2/bdavis/sites/ms/pyccd-results/H19V14/2017.08.18/json >& H19V14_json_inventory.txt

Estimated Time (h:mm:ss):

1:05:00

Step 3: Create Visual Change Map Products

Scripts Used:

- change_maps.sh
- api.py
- change_maps.py
- change_products-cli.py
- change_products.py
- class_maps.py
- class_products.py
- commons.py
- compress_json.py
- density.py
- geo_utils.py
- json_matlab.py
- logger.py
- matlab-cli.py

Summary:

change_maps.sh is a parent bash script to receive, check and parse arguments to the python client application change_products-cli.py. Annual visual change maps products (.tif files) are produced on a per-tile basis for:

- Change Magnitude
- Last Change
- Segment Length
- QA

An overall coverage map is also produced.

change_maps.sh <site> <version> <h> <v>

The python scripts are obtained from the git repository:

https://github.com/klsmith-usgs/json-matlab.git

Example Syntax

change_maps.sh ms 2018.08.18 19 14

Estimated Time (h:mm:ss):

0:01:15

Total Estimated Time for Steps 1-3 (h:mm:ss):

4:20:00

Step 4: Optionally Compare Results from Different Versions

compare_json_results.py (under development)

All keys and values from 2 json files are read and compared to ensure mathematical value differences fall within a numerically accurate tolerance value. Any numerical value difference outside the tolerance value, along with any non-numerical field differences will be written to stdout.

python compare_json_results.py < version-1-json file > <version-2-json-file> <tolerance-value>

Example Syntax

python count_json_results.py H19V14_284415_1214805_207108.18.json H19V14_284415_1214805_2017.10.27.json 0.000001

Estimated Time (h:mm:ss):

0:00:0