CSPP VIIRS SDR Performance

CSPP Team: Nick Bearson, Geoff Cureton, Jim Davies, Ray Garcia, Liam Gumley, Graeme Martin, Scott Mindock, Nadia Smith, Kathy Strabala, Elisabeth Weisz

Scott.Mindock@ssec.wisc.edu

CSPP SDR Topics

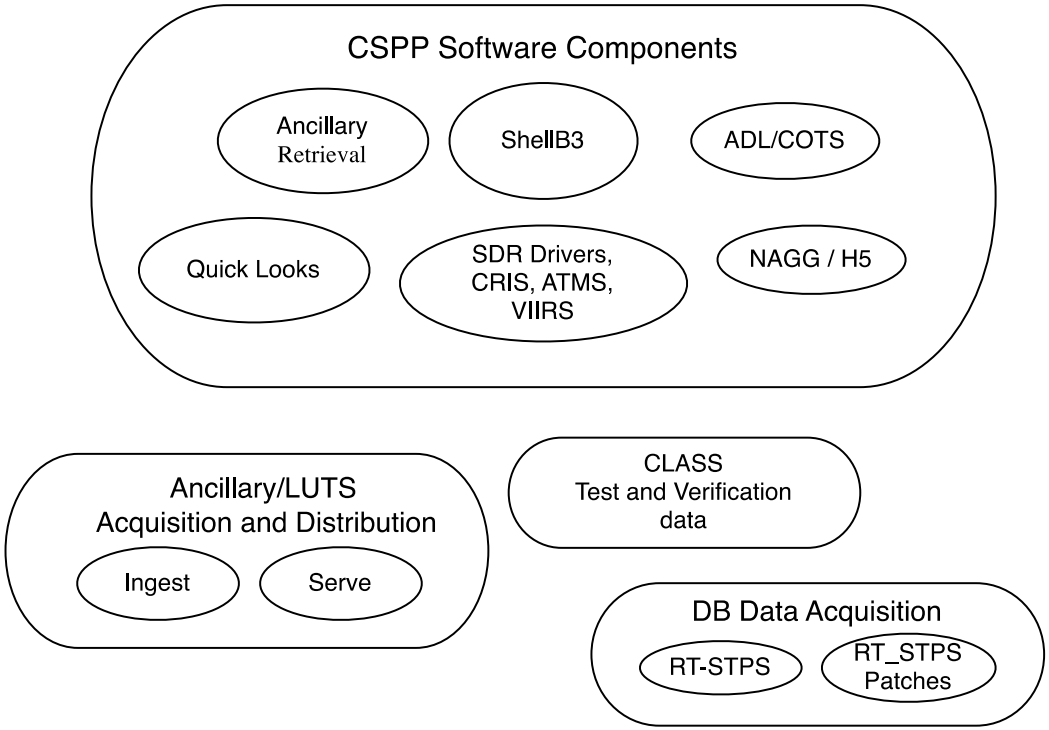
- CSPP Components
- What is Performance?
- Performance Options
- CSPP VIIRS SDR Changes
- Results
- Conclusions

CSPP SDR Components

More than just a pretty processing package. Acquisition, Ancillary and Verification

Todays focus is on the SDR Drivers, RT-STPS and CLASS

Past focus on ease of use and reliability



Performance Measures/Bottlenecks

- File I/O
- Memory
- CPU
- Network

What are we improving?

Reduce time from Direct Broadcast data availability to SDR products.

What is best case?

RDR creation time + SDR creation time

The end point can be all SDRs or a single SDR

Performance Improvement Options

- ADL Code Changes
- Compiler Updates
- Parallelism Options
- Pipelining Options

Options for Improvements

- Change the code. Modify ADL VIIRS SDR, analysis shows math libraries and geo-location code. (Not practical)
- Change the tools. CSPP 1.3 based on GCC 4.5 tool chain. CSPP 1.4 based on GCC 4.7
- Change the driver. Use Granule based parallelism. Leverage PYTHON multi-processing package
- Change the flow. Modify RT-STPS to pipeline the data acquisition process.

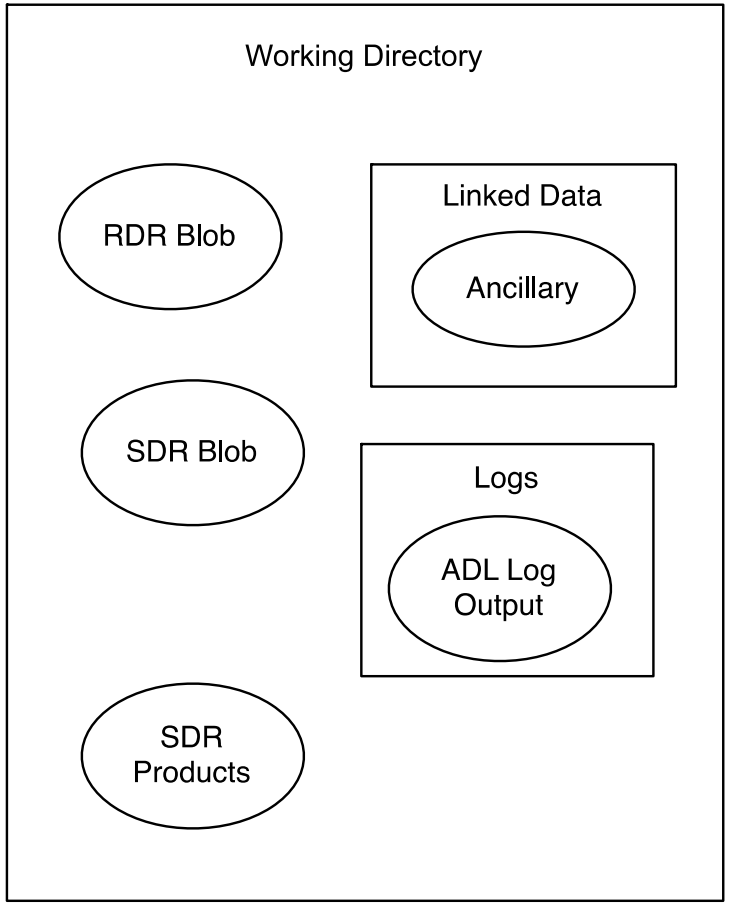
Preparation

- Refactoring of CSPP VIIRS SDR driver.
- Changes to working directory structure.
- Knowledge of RT-STPS NPP RDR generation.

CSPP VIIRS SDR 1.3 Working Directory

Flat directory,
Blobs and products
stored together

ADL may experience
race condition issues

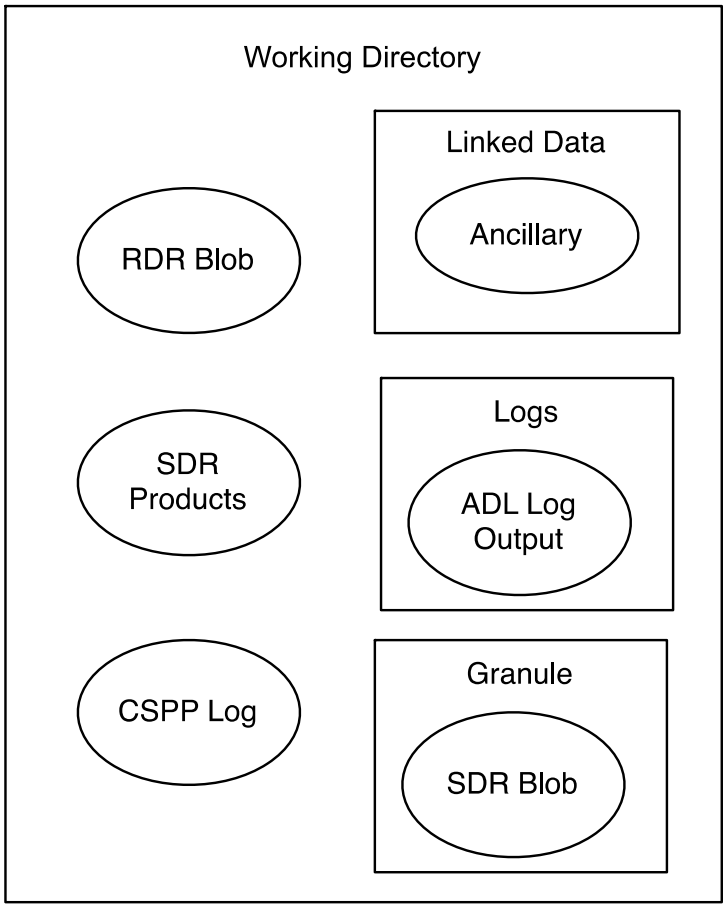


CSPP VIIRS SDR 1.4

Granule based sub-directories introduced.

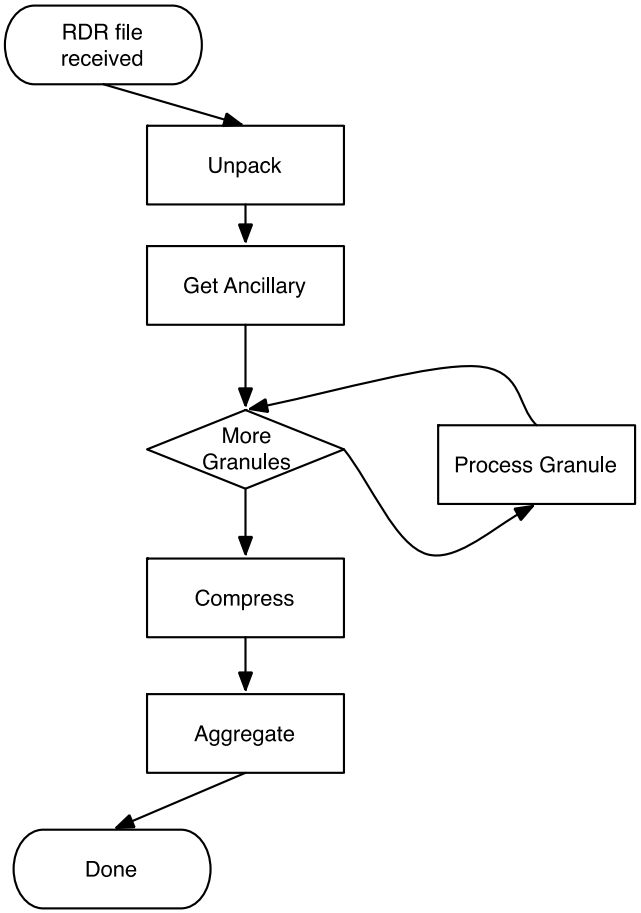
Intermediate and SDR blobs stored in granule based sub-directory

Processing log saved on error conditions.



CSPP VIIRS SDR 1.3

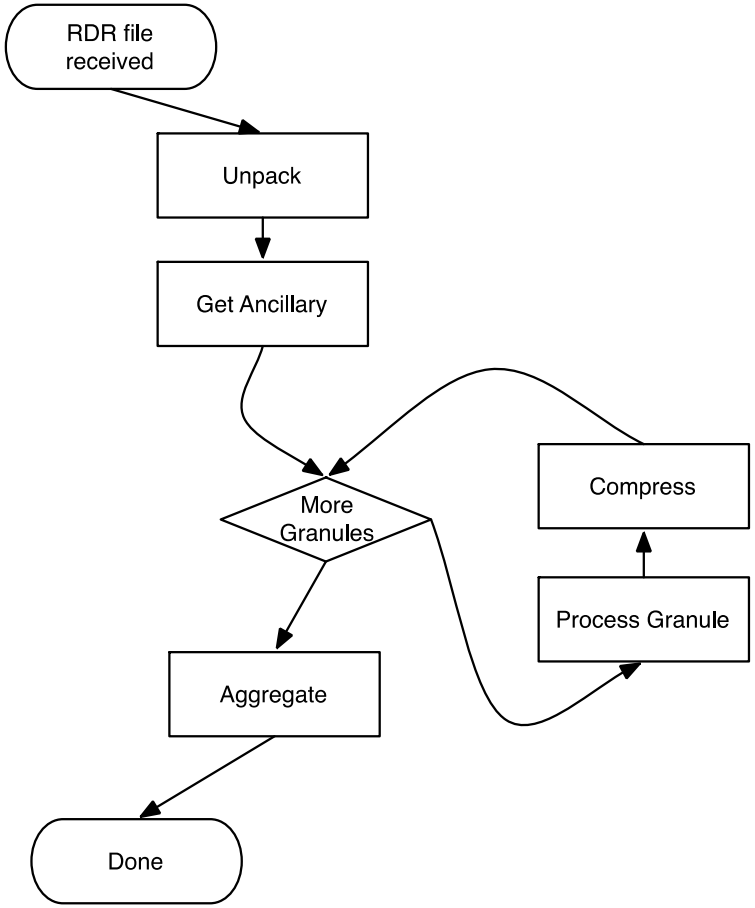
- Processing is single threaded.
- Processing loops through granules.
- Processing loops through compression.
- Results aggregated.



CSPP VIIRS SDR 1.4

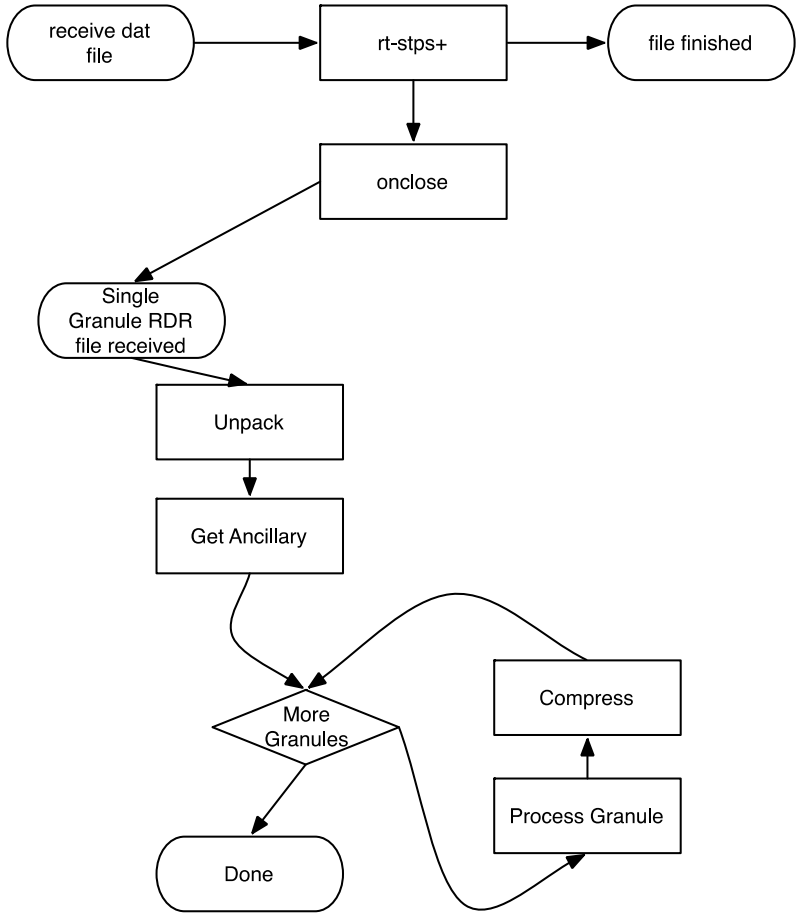
- Python multi-processing is leveraged.
- Granule creation and compression combined.
- Aggregation remains single threaded.
    - Multi-processing runs ADL followed by compression.
    - Number of processors specified on command line.

Viirs_sdr.sh –p 4 -z



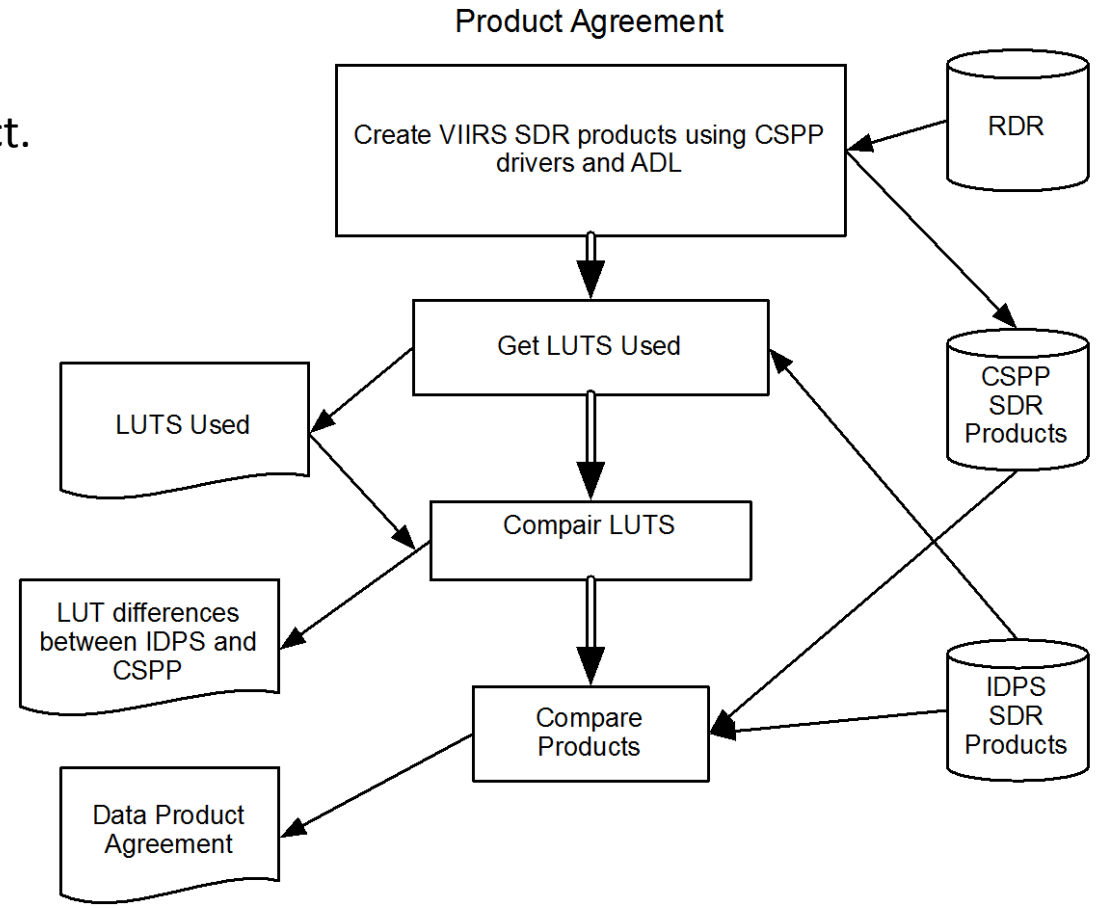
CSPP VIIRS SDR 1.4 +

- RT-STPS modified to produce separate RDR file for each granule.
- Onclose.py routine used to invoke viirs_sdr.sh
- Viirs_sdr.sh run once for each RDR file.
- Working directory is shared
- Aggregation is gone



Did I break it?

- CSPP VIIRS SDR products compared to IDPS product.
- LUTS used are compared.
- Data sets are compared.



Did I break it?

- CSPP VIIRS SDR products created using several processing options.
- Single and Multiprocessor versions of products compared.
- Used h5diff with machine tolerances.
- No – I do not think I broke it.

Results

- Test Machines
- Compiler Changes
- Parallelism
- Pipelining

Machine Properties

Name	Brand	Cpu MHz	Cache	Cores
JPSS-CLOUD	AMD Opteron(tm) Processor 6176	2300.032	512 KB	12
DRAGON	Intel(R) Xeon(R) CPU E5-2690 0 @ 2.90GHz	2899.735	20480 KB	8

Compiler Differences, 5+1 Granules

Name	CSPP VER	Real Min:Sec	System Sec	User Sec	Total Sec	Granule Sec
JPSS-CLOUD	1.3	53:14.0	2124.32	1797.11	3921.43	784.29
JPSS-CLOUD	1.4	59:10.8	2503.16	1079.48	3582.64	716.53
DRAGON	1.3	08:09.7	37.39	1270.91	1308.30	261.66
DRAGON	1.4	07:46.5	38.86	479.58	518.44	103.69

This is the difference between GCC 4.5 and 4.7
8.6% to 60% Improvement

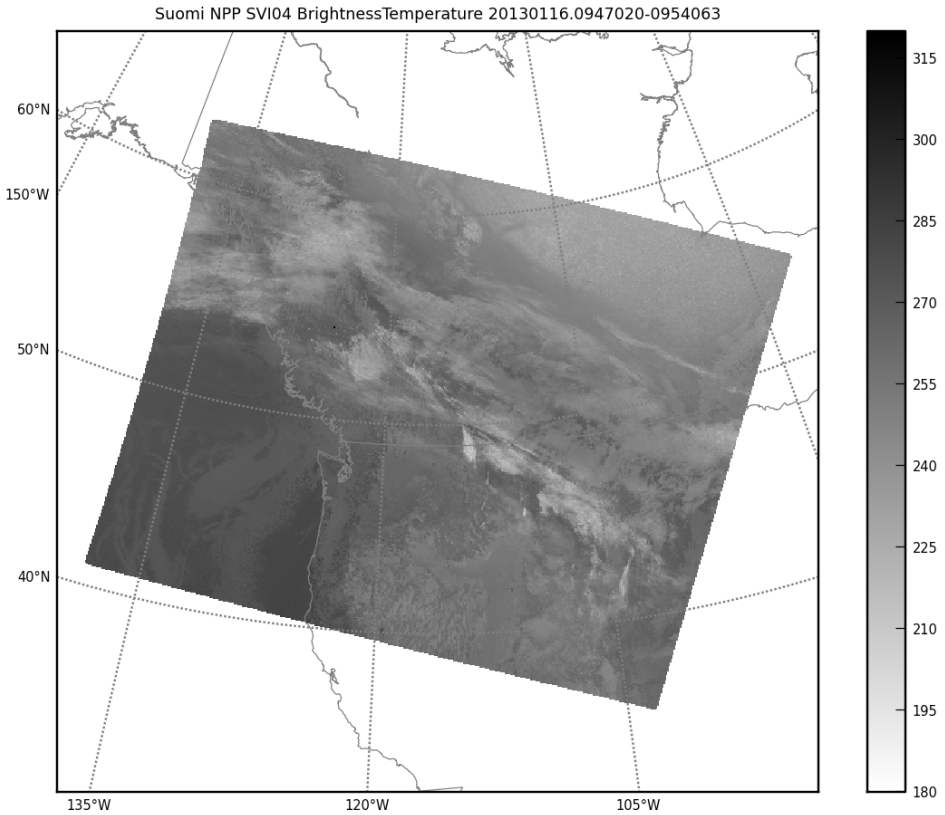
Multiprocessing, 5+1 Granules

Name	CSPP VER	Cores	Real Min:Sec	System Sec	User Sec	Total Sec	Granule Time Sec
JPSS-CLOUD	1.4	1	59:10.8	2503.16	1079.48	3582.64	716.53
		2	33:07.5	2302.60	1077.10	3379.70	675.94
		4	19:15.0	2044.96	1081.75	3126.71	625.34
		6	10:05.4	1323.88	1080.71	2404.59	480.92
DRAGON	1.4	1	07:46.5	38.86	479.58	518.44	103.69
		2	04:36.3	37.14	474.56	511.70	102.34
		4	03:08.3	40.90	483.11	524.01	104.80
		6	01:43.0	45.96	478.27	524.23	104.85

Pipelined

- GRAN_START: Mon May 20 21:12:22 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0946086_e
- GRAN_END: Mon May 20 21:12:24 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0946086_e
- GRAN_START: Mon May 20 21:12:48 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0947020_e
- GRAN_START: Mon May 20 21:13:13 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0948256_e
- GRAN_START: Mon May 20 21:13:37 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0949510_e
- GRAN_START: Mon May 20 21:13:47 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0951164_e
- GRAN_START: Mon May 20 21:13:47 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0952418_e
- GRAN_END: Mon May 20 21:13:50 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0951164_e
- GRAN_END: Mon May 20 21:14:33 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0947020_e
- GRAN_END: Mon May 20 21:14:56 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0948256_e
- GRAN_END: Mon May 20 21:15:13 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0949510_e
- GRAN_END: Mon May 20 21:15:44 UTC 2013 File: RNSCA-RVIRS_npp_d20130116_t0952418_e

Pipelined Quicklook



Conclusions

- Keep your tools up to date.
- Try simple solutions before attempting more complicated options.
- High performance can be achieved by a combination of improvements.