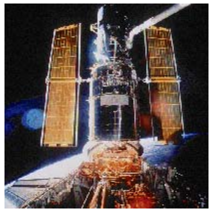
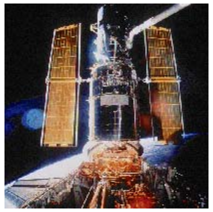


HTCondor Use In Operational Data Processing For The Hubble Space Telescope (HST) And The James Webb Space Telescope (JWST)

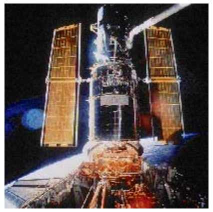
Michael Swam, Mary Romelfanger,
with acknowledgement to former team member and lead designer Francesco Pierfederici



Who Are We (STScI) ?

- The Space Telescope Science Institute (STScI) is located in Baltimore, Maryland on the campus of Johns Hopkins University
- STScI (est. 1981) houses employees of AURA, ESA, and CSRA
- AURA – Association of Universities for Research in Astronomy
  - A non-profit consortium of universities dedicated to astronomy research
  - Holds NASA contracts for mission operations, science operations, or aspects of those, for various space-based observatories
  - Runs several ground-based observatories (Gemini)
- ESA – European Space Agency
  - Various international partner agreements define work roles on the NASA contracts
- CSRA – formerly Computer Sciences Corporation (sub-contractor)

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



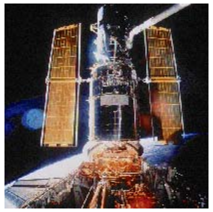
Astronomy Missions at STScI (mostly space observatories)

- Hubble Space Telescope (HST) : Science Operations, 1990
- James Webb Space Telescope (JWST) : Mission + Science Operations, 2018
- Kepler : Initial Data Processing and Archive, 2009
- Transiting Exoplanet Survey Satellite (TESS) : Archive, 2017
- Wide-Field InfraRed Survey Telescope (WFIRST) : Science Operations (shared with IPAC, GSFC), 202x
- *MAST – Mikulski Archive at Space Telescope : Multi-Mission Archive holding HST, Kepler, Galex, FUSE, Astro-1,2 (HUT,UIT,WUPPE), etc.*
- This talk will focus on the 2 main missions, so far: HST and JWST
  - One uses HTCondor already, and one will very soon

(Are you getting acronym overload yet?

WARNING: This is NASA, the barrage has just begun…)

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



HST (aka "The Hubble")

- Launched: April 1990 from Space Shuttle Discovery
- Serviced: 5 times, last in 2009
- Low-Earth Orbit: ~ 570 km
- Size comparison: a school bus
- Original (retired) instruments:

  FOC, FOS, GHRS, HSP, WFPC-1
- 2nd generation retired instruments:

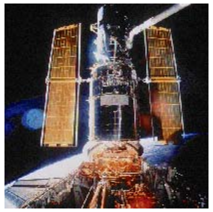
  COSTAR, WFPC-2
- "Active" instruments:

  ACS, COS, FGS, NICMOS, STIS, WFC3
- End of life: 202x ?

  -de-orbit mission?

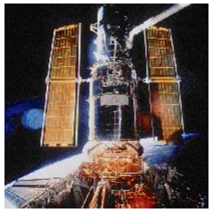
  -if not atmospheric drag will eventually win and the mirror will land *somewhere…*

STScI | SPACE TELESCOPE SCIENCE INSTITUTE

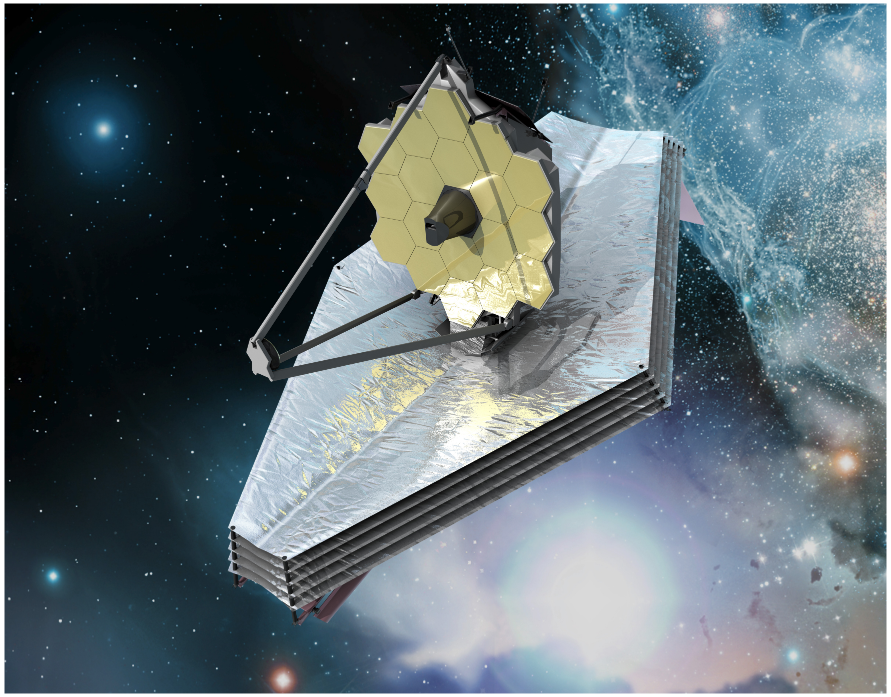


HST Data Processing – Short History

- Original data processing system (PODPS) delivered to STScI under NASA contract from a vendor (TRW) to support launch/science ops in 1990
- A redesigned data processing system (OPUS) developed by STScI went operational in 1995, and is still used for some aspects of HST data processing today
- OPUS features
  - Distributed processing on a pool of machines (shared disk access)
  - Uses a blackboard paradigm for distributing work
    - Competing processes scan blackboard data structure, looking for work requests they can satisfy, attempt to grab a job, only 1 will win the job, mark the blackboard entry as taken, blackboard entry updated on success/failure
  - Originally used file-system blackboards (directories of empty files)
  - Redesigned in 2002 to use in-memory blackboards, held in servers communicating via CORBA
- WHY change?

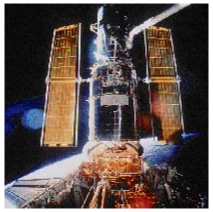


JWST ("The Webb" isn't going to stick…)



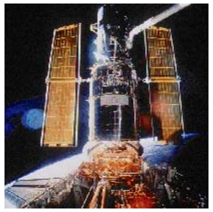
- Planned Launch: Fall 2018 from ESA Spaceport, French Guiana, on an Ariane5 rocket
- Servicing plans: None, due to cost (but I thought I heard they included a grappling handle…)
- Orbit: Solar at 2nd Lagrange point (L2), beyond the moon, 1.5 <u>million</u> km away
  - Why? Continuous viewing + stable thermal environment
- Size comparison: a tennis court
- Instruments: FGS, NIRCAM, NIRISS, NIRSPEC, MIRI
- End of life: 5.5 yrs planned, goal = 10+ yrs, limiting factor = propellant for orbit station keeping

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



HST / JWST : The Future of OPUS

- OPUS was re-evaluated under a trade-study in 2011 to choose the "Workflow Management System" (WMS) for the James Webb Space Telescope Data Processing System
- When examining current technologies and trade-offs, HTCondor came to the fore, and was selected for use by JWST, after a prototyping period and add-on development was performed
  - Added STScI's "Orchestrated Workflow Layer" = OWL
- The HST Mission Office decided to allow OPUS to be phased out for HST Data Processing, in favor of HTCondor/OWL
  - HST wins by gaining a more maintainable and versatile data processing platform to meet HST's data processing needs until the end of mission, i.e. a technology "refresh"
  - JWST wins by gaining a well-vetted Workflow Management System with years of use by an operational mission (HST) by the time JWST launches
  - Data Processing operations team wins by mastering just 1 system

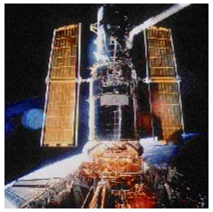


Why the move to HTCondor?

- Better performance/flexibility for large processing runs
  - Pool our TEST and OPS machines, at least when active testing is not occurring
  - Capability to add machines when needed, then release them for normal ops loads
- A more maintainable, reliable system into the future
  - OPUS was developed in-house, but the expertise has left
  - Uses highly-detailed C++/CORBA (not a lot of developers find this attractive), dependence on 3rd-party CORBA ORB (ACE/TAO)
  - Debugging complexity (usually just restart the CORBA servers + pipelines)
  - OPUS development has been rusty-railed for many years now, though it's long lifetime and continued use is testament to the solid design
  - Operating system maintenance, network security additions make this an increasing risk
  - HUGE HTCondor user base, community of expertise and applications

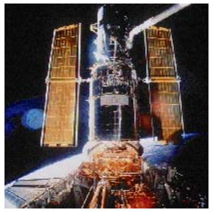
Why is OWL necessary?

- OWL = Orchestrated Workflow Layer
  - We found that HTCondor manages compute resources extremely well, but lacked services for managing and tracking the data being processed
  - Our experience with OPUS and HST Data Processing convinced us of the need to manage and track the data flows through the computers over time
- What does OWL provide?
  - A job-tracking database table that captures the full ClassAd plus a few extras, for every workflow job step run by the system
    - Populated and updated via HTCondor job hooks
      - still not sure if that was a good choice; fine when our hooks work, hard to diagnose when they fail (unable to capture stderr?)
  - Template-driven workflow generation ("DAGs on-the-fly") using the Jinja2 template engine
  - A web app (OWL GUI) for monitoring dataset processing status and other system features



More OWL add-ons

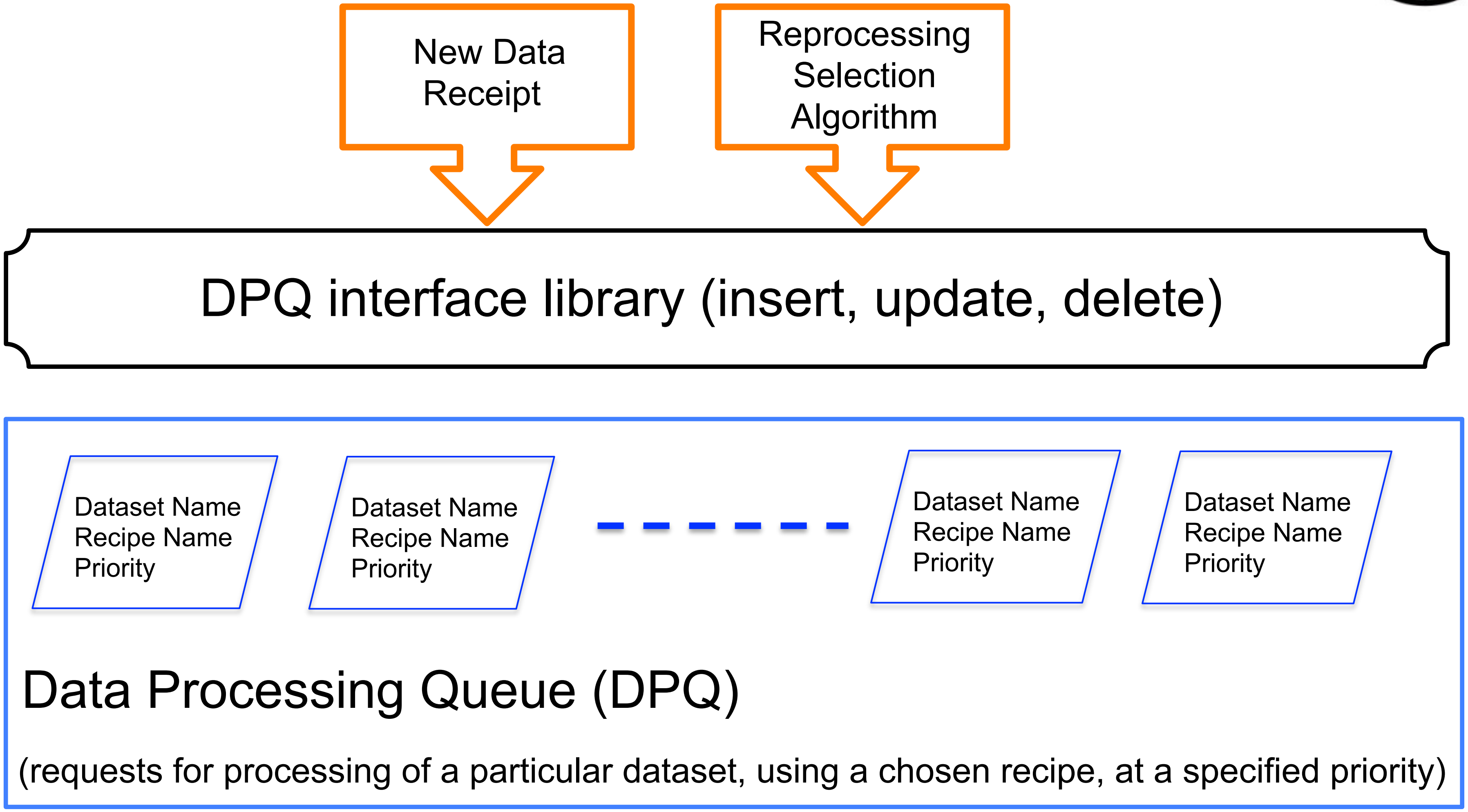
- We also wanted a way to specify and feed data processing runs into the system
  - Created a simple Data Processing Queue (DPQ) database table to hold workflow requests that include
    - Dataset name
    - Workflow type
    - Relative priority
  - Created "The Shoveler" task to govern the order and rate at which DPQ entries are sent through OWL workflow template generation to be transformed into executing DAGMan jobs on the HTCondor pool
  - Created a re-usable, configurable Poller task that can be instantiated to watch for the arrival of certain files in a directory and turn each "file event" into a set of DPQ inserts, to request data processing for that event
- Added a user-specific rescue Server that receives OWL GUI rescueDAG requests to re-try a failed workflow



What do we accomplish with this system?

- Our paradigm: process everything once, reprocess MANY times…
- Science data is processed once on initial receipt from the observatory
  - Confirms that the instrument mode can be successfully processed
  - Saves the data products in the archive + safestore (media taken offsite)
  - Populates the archive catalog (database) of searchable metadata to support archive research discovery
  - Data accounting metrics are collected to verify completeness and link back to the observatory planning system to verify the proposer got all the observations they asked for
- Reprocessing is driven by improvements in
  - Calibration reference data (darks, flats, distortion models, etc.)
  - Calibration algorithm improvements
  - Metadata enhancements/fixes (give more complete/useful information related to the science observations to aid in interpretation and analysis)
  - Datasets are reprocessed ~5-30 times over the mission lifetime

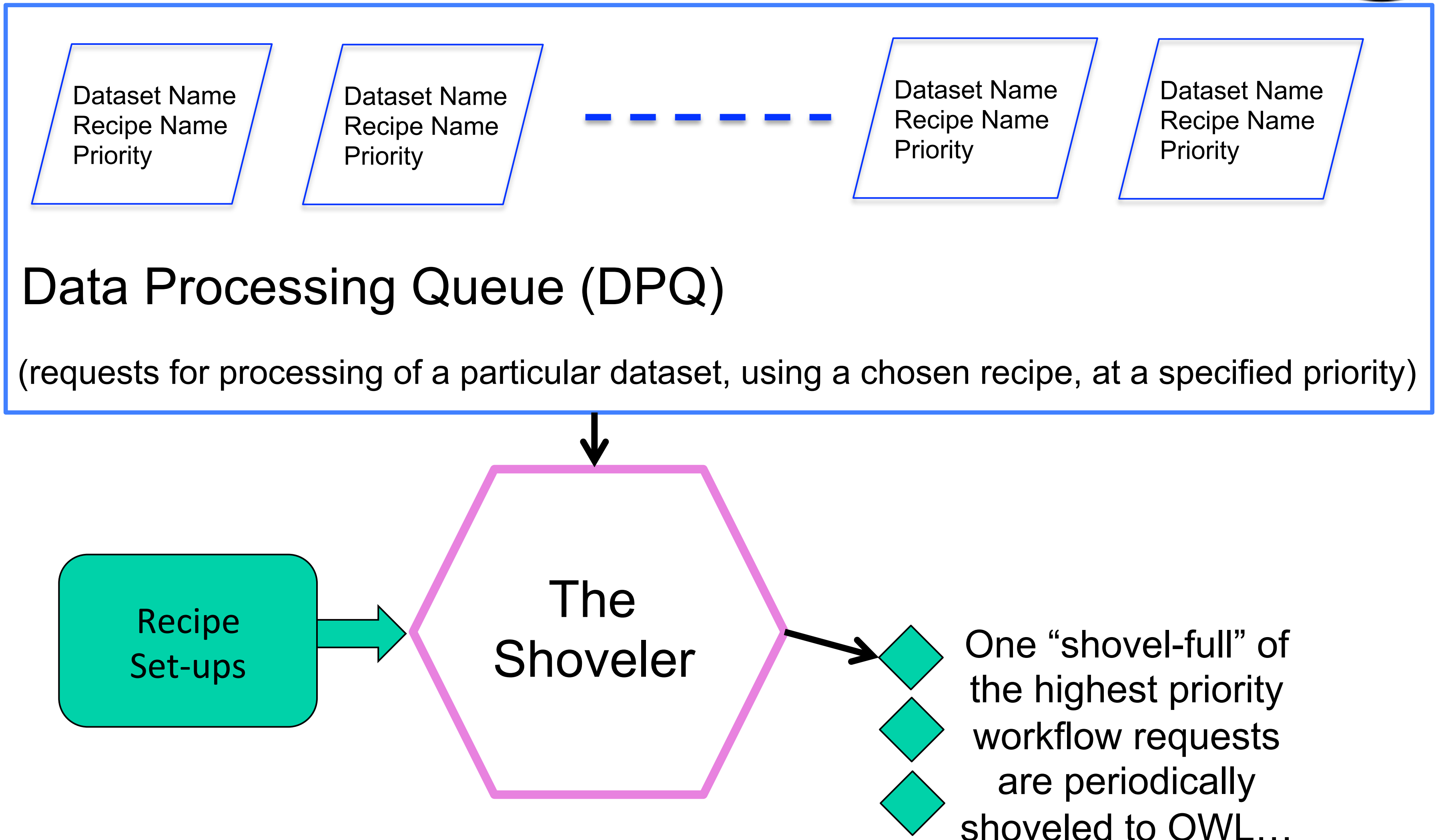
The Flow



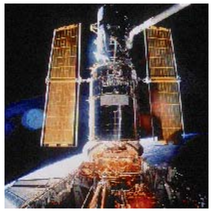
(requests for processing of a particular dataset, using a chosen recipe, at a specified priority)

SPACE TELESCOPE SCIENCE INSTITUTE

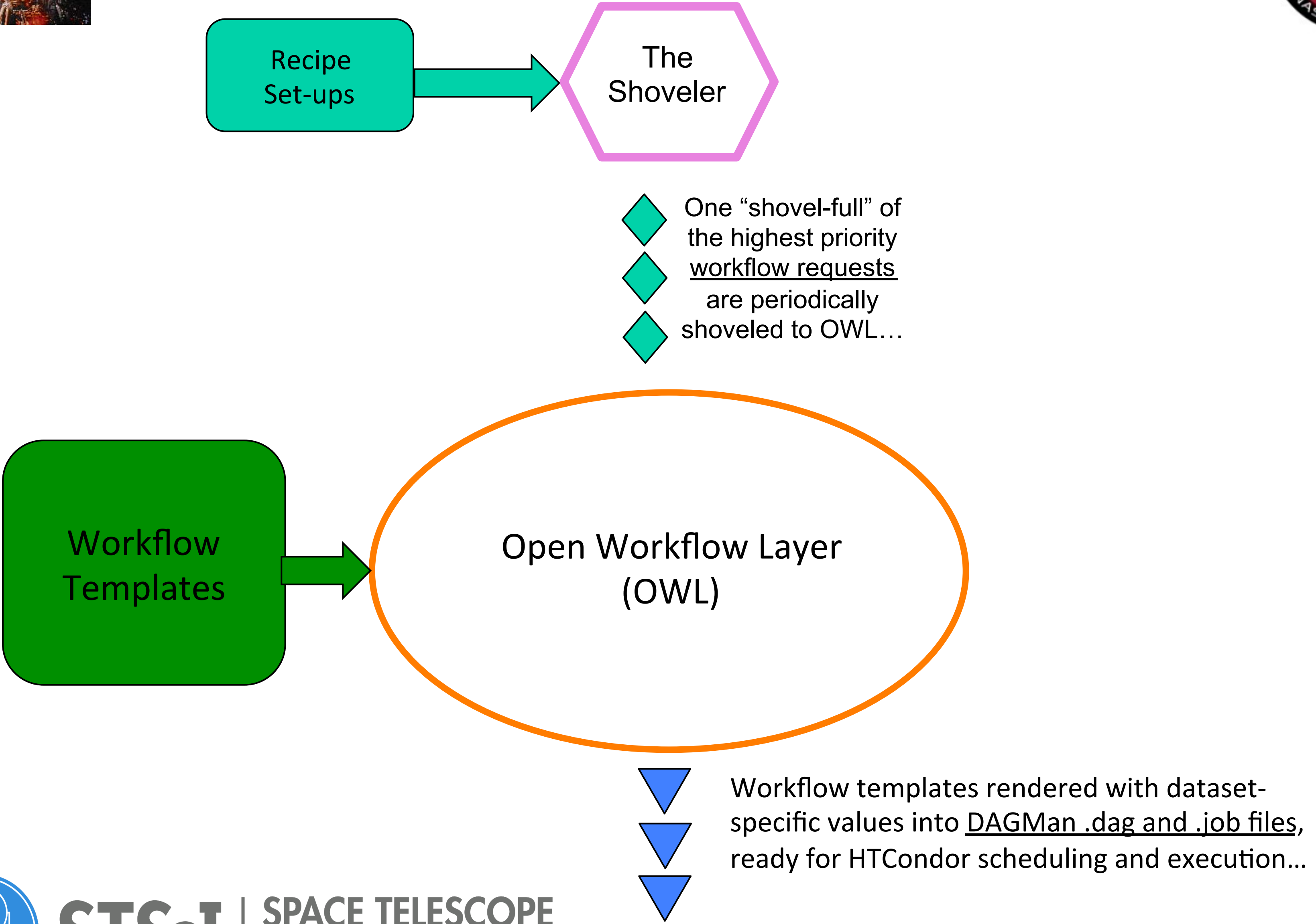
Data Processing Queue (DPQ) and the Shoveler



Dataset Name
Recipe Name
Priority
Dataset Name
Recipe Name
Priority
Dataset Name
Recipe Name
Priority
Dataset Name
Recipe Name
Priority
Data Processing Queue (DPQ)
(requests for processing of a particular dataset, using a chosen recipe, at a specified priority)
Recipe Set-ups
The Shoveler
One "shovel-full" of the highest priority workflow requests are periodically shoveled to OWL…



OWL Translates Workflow Request Into .dag/.job files


Recipe Set-ups
The Shoveler
One "shovel-full" of the highest priority workflow requests are periodically shoveled to OWL…
Workflow Templates
Open Workflow Layer (OWL)
Workflow templates rendered with dataset-specific values into DAGMan .dag and .job files, ready for HTCondor scheduling and execution…

Workflows (.dag/.job) run by HTCondor

Workflow Templates → Open Workflow Layer (OWL)

Workflow templates rendered with dataset-specific values into DAGMAN .dag and .job files, ready for HTCondor scheduling and execution.
Submitted via DRMAA v1 (www.drmaa.org)

Distributed Resource Mgmt Application API (DRMAA v1)

HTCondor
High Throughput Computing

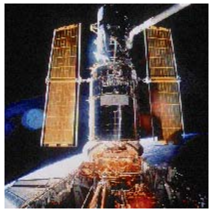
Local Processing Cluster

Small Auxiliary Resources

Desktop
“cycle scavenging”

Academic, Commercial, or Government Cloud-computing facilities

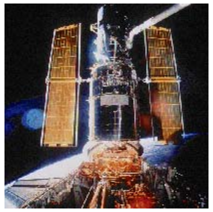
STScI | SPACE TELESCOPE SCIENCE INSTITUTE



Example DAG template

```
#-------------------------------------------------------------------------------
# crds_delivery.dag
#
# Handles processing of a catalog file from the CRDS team, found by the CRDS
# instantiation of the ManifestFilePoller
#
# 2015-07-24 75701  LThompson first version for JWST
# 2015-09-28 80484  Mary      added PRIORITY
#-------------------------------------------------------------------------------

# Job definitions
JOB CRDS_SUBMIT crds_submit_{{ dataset }}.job
JOB CRDS_CONFIRM crds_confirm_{{ dataset }}.job

# Job Relationships
PARENT CRDS_SUBMIT CHILD CRDS_CONFIRM

PRIORITY CRDS_SUBMIT    180
PRIORITY CRDS_CONFIRM   190
```

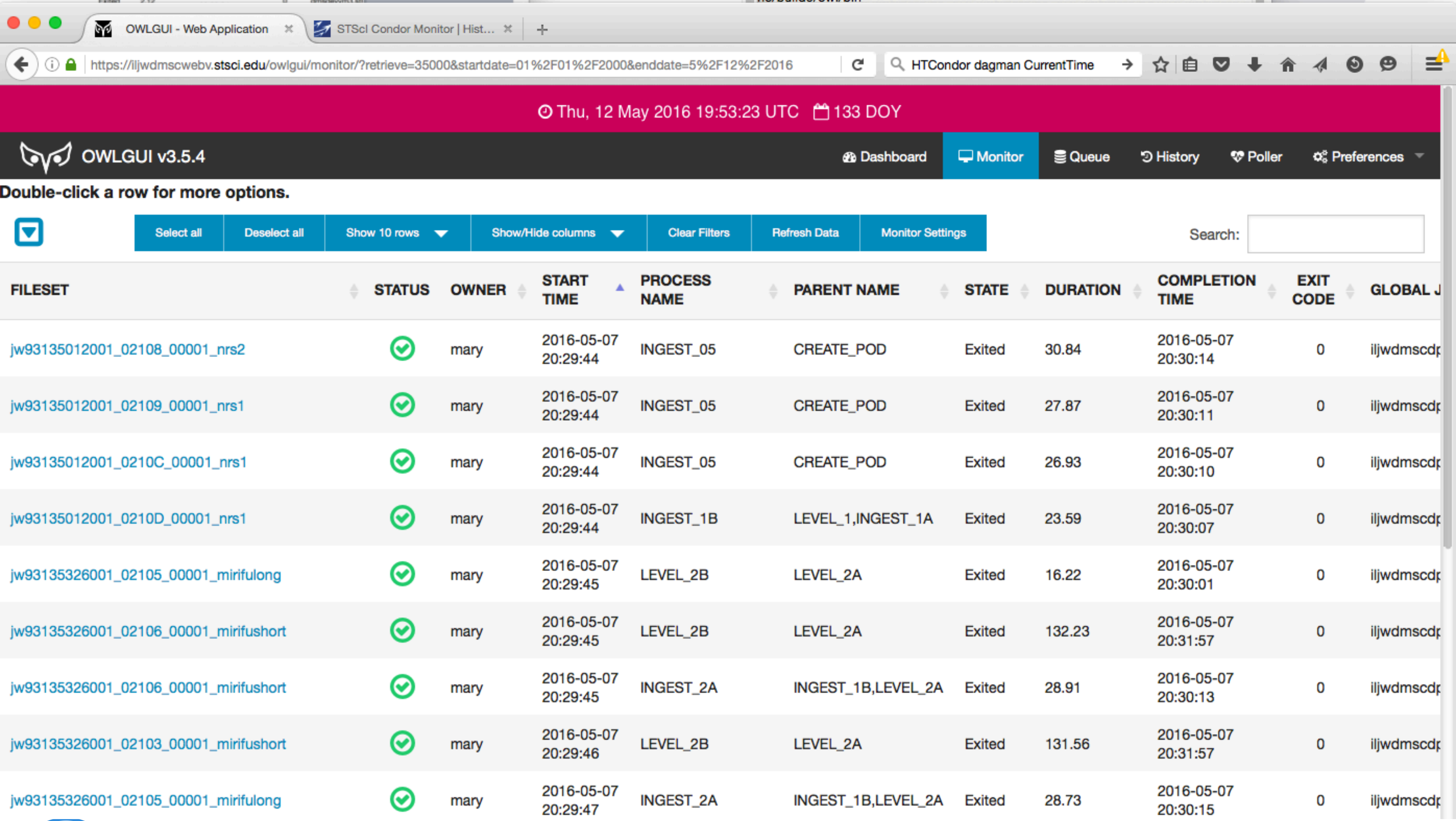
SPACE TELESCOPE SCIENCE INSTITUTE



Example JOB template

```
#-----------------------------------------------------------------------------
stage_name = crds_submit
Executable = $ENV(PIPELINE_BASE)/bin/CrdsFileDelivery.py
Arguments = -r crds_delivery -p {{pathname}} {{dataset}} submit

log_stamp = $$([CurrentTime])
+Out       = "ALOG_$(log_stamp)_$(stage_name)_{{ dataset }}.out"
+Err       = "ALOG_$(log_stamp)_$(stage_name)_{{ dataset }}.err"

Universe = vanilla
Log = {{ dataset }}.condor_log

+InputDataset = "{{ dataset }}"
+HookKeyword = "OWL"
+Instances = 1

getenv = True
Notification = Never
Queue
```

OWL GUI – Monitor Pane

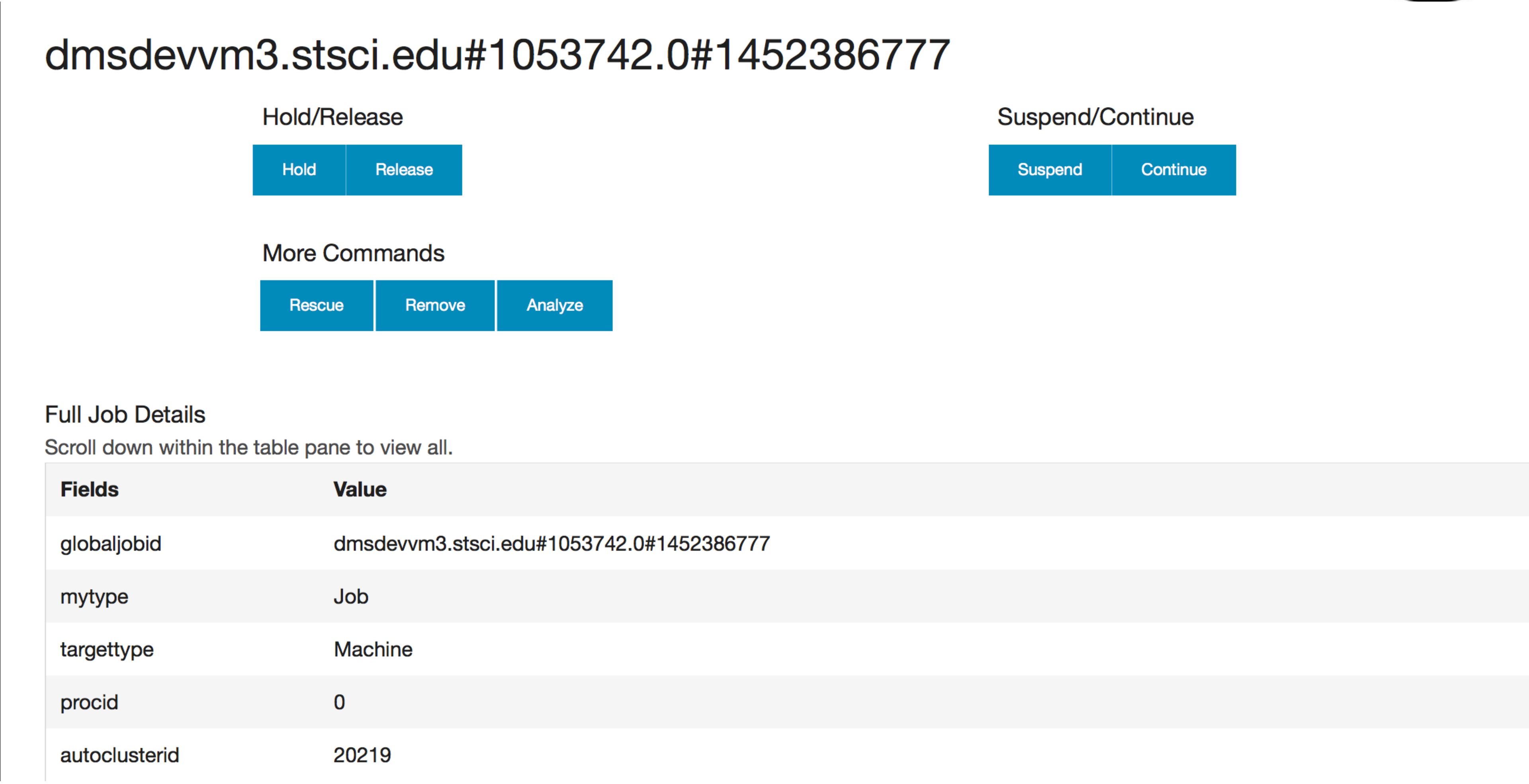


OWLGUI - Web Application
STScI Condor Monitor | Hist...
https://iljwdmscwebv.stsci.edu/owlgui/monitor/?retrieve=35000&startdate=01%2F01%2F2000&enddate=5%2F12%2F2016
HTCondor dagman CurrentTime
Thu, 12 May 2016 19:53:23 UTC
133 DOY
OWLGUI v3.5.4
Dashboard
Monitor
Queue
History
Poller
Preferences
Select all
Deselect all
Show 10 rows
Show/Hide columns
Clear Filters
Refresh Data
Monitor Settings
Search:

Double-click a row for more options.

FILESET	STATUS	OWNER	START TIME	PROCESS NAME	PARENT NAME	STATE	DURATION	COMPLETION TIME	EXIT CODE	GLOBAL J
jw93135012001_02108_00001_nrs2	✓	mary	2016-05-07 20:29:44	INGEST_05	CREATE_POD	Exited	30.84	2016-05-07 20:30:14	0	iljwdmscdp
jw93135012001_02109_00001_nrs1	✓	mary	2016-05-07 20:29:44	INGEST_05	CREATE_POD	Exited	27.87	2016-05-07 20:30:11	0	iljwdmscdp
jw93135012001_0210C_00001_nrs1	✓	mary	2016-05-07 20:29:44	INGEST_05	CREATE_POD	Exited	26.93	2016-05-07 20:30:10	0	iljwdmscdp
jw93135012001_0210D_00001_nrs1	✓	mary	2016-05-07 20:29:44	INGEST_1B	LEVEL_1,INGEST_1A	Exited	23.59	2016-05-07 20:30:07	0	iljwdmscdp
jw93135326001_02105_00001_mirifulong	✓	mary	2016-05-07 20:29:45	LEVEL_2B	LEVEL_2A	Exited	16.22	2016-05-07 20:30:01	0	iljwdmscdp
jw93135326001_02106_00001_mirifushort	✓	mary	2016-05-07 20:29:45	LEVEL_2B	LEVEL_2A	Exited	132.23	2016-05-07 20:31:57	0	iljwdmscdp
jw93135326001_02106_00001_mirifushort	✓	mary	2016-05-07 20:29:45	INGEST_2A	INGEST_1B,LEVEL_2A	Exited	28.91	2016-05-07 20:30:13	0	iljwdmscdp
jw93135326001_02103_00001_mirifushort	✓	mary	2016-05-07 20:29:46	LEVEL_2B	LEVEL_2A	Exited	131.56	2016-05-07 20:31:57	0	iljwdmscdp
jw93135326001_02105_00001_mirifulong	✓	mary	2016-05-07 20:29:47	INGEST_2A	INGEST_1B,LEVEL_2A	Exited	28.73	2016-05-07 20:30:15	0	iljwdmscdp

OWLD interface to HTCondor + ClassAd



dmsdevvm3.stsci.edu#1053742.0#1452386777

Hold/Release

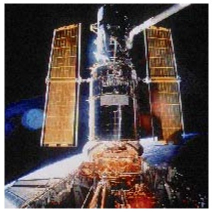
Suspend/Continue

More Commands

Full Job Details

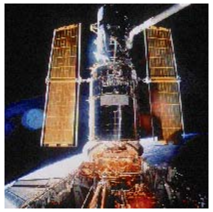
Scroll down within the table pane to view all.

Fields	Value
globaljobid	dmsdevvm3.stsci.edu#1053742.0#1452386777
mytype	Job
targettype	Machine
procid	0
autoclusterid	20219



Operations View

- Nearly all of the previous steps are automatic
- Operators inspect the OWL GUI “Monitor” tab and filter for failed job steps
  - E.g. science calibration software can fail if matching calibration reference data is not found on disk for the dataset’s instrument configuration
- OWL GUI can display log files showing the details on each failure
- Operations can take corrective action outside the system
  - E.g. contact the instrument science team to request additional calibration reference data
- Operations uses the OWL GUI to invoke a rescueDAG and re-runs the failed workflow to get past the original failure
- OWL GUI provides “Poller” and “DpQueue” tables for alternate views

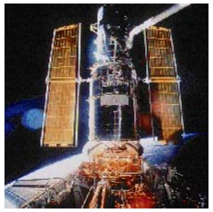


HST – Daily Data Rates (very small)

- Average downlink volume for science telemetry files
  - 4.7 GB per day   [2013, 2014]
- 4 active science instruments
  - ACS, WFC3: mainly imaging
  - COS, STIS: mainly spectroscopic
- On avg. 155 new science observations arriving each day
  - Average mix of observations each day
    - ACS = 22, COS=10, STIS= 46, WF3 =77 [2013,2014]

      (14%   6%   30%   50%   )
  - Approx. 2/3 imaging, 1/3 spectroscopic
- Average archive ingest rate
  - 17.1 GB per day (data 'expansion' rate ~ 4x)

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



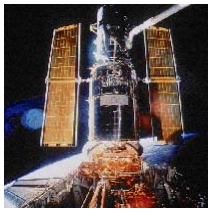
HST Datasets in Archive (for reprocessing)

- Current number of science datasets for each active (repro) instrument

– ACS:	222849
– COS:	32698
– STIS:	172970
– WFC3:	213374

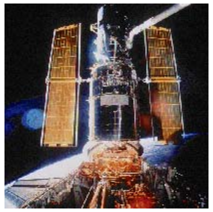
- A full-instrument reprocessing can require >200K observations to be sent back through the processing algorithms to generate new data products for re-ingest into the archive (old versions are "invisible")
- All reprocessing runs at a lower priority than new downlinked data and special processing (e.g. calibration reference file updates)
- The Shoveler honors the priorities in submitting work to HTCondor
- .job steps are also given priorities in the .dag files, so that once they are running on the HTCondor pool, they proceed in relative priority-step order

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



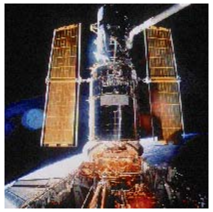
A Reprocessing Scenario

- Reprocessing is driven by improvements to calibration reference data or software
  - E.g. Calibration programmers develop a calibration software improvement that would result in improved science data products
- Database queries are run against the archive catalog (database tables describing the characteristics of every science exposure held in the archive) to identify datasets that could benefit from the improvement
- Dataset names and the appropriate type of reprocessing workflow are submitted to the Data Processing Queue, at lower priority than workflows for “fresh” data downlinks from the observatory
- The Shoveler picks up the reprocessing entries when the priorities allow, submits them through OWL to HTCondor and the improved science data products are generated and submitted to the data archive
- The system handles a mix of new data from the observatory and reprocessing scenarios improving data already in the archive



Hardware – HST Ops (very small)

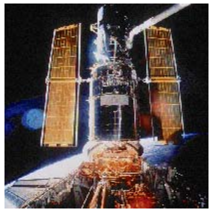
- Currently 7 physical machines for daily processing / reprocessing
  - OpsLittleHelper – COLLECTOR, NEGOTIATOR, MASTER
    - 2 cores, each w 1Gb memory
  - Ops1,2,3,4,5,6 – SCHEDD, STARTD, MASTER, OWLD
    - 160 cores, each w/6Gb memory
    - 288 cores, each w/2Gb memory
    - 450 TOTAL cores
  - Red Hat Enterprise Linux 5
    - far behind, we know! Due to a C++/database interface layer dependency we are working through
- Shared disk access over 10Gb network to Isilon Network Attached Storage (NAS)
- MS SQL Server databases on a clustered auto fail-over set of Win2008R boxes
- 1Gb network to other machines
- Just Added: CycleServer (www.cyclecomputing.com) for pool monitoring and management



HTCondor becomes operational for HST

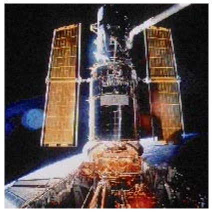
- Dec-2013: Initially installed in a single non-critical processing area (parallel calibration reference file delivery and archiving)
  - Ops gained comfort/experience in operating HTCondor/OWL, while still relying on OPUS for initial science data processing and reprocessing
  - Further development on OWL and HST science workflows was needed before science processing was ready for operational use
- Dec-2014: Moved reprocessing for active instruments over to HTCondor/OWL
  - June 2015: 15230 COS data products processed in 66 hrs
  - Sept 2015: 90959 WFC3 data products processed in ~15 days
  - Dec 2015: 16156 COS products processed in just over 52 hrs
- Oct-2015: First-look data workflows added
  - All new data downlinked from HST is processed by HTCondor/OWL
- TBD:
  - Full ACS instrument collection reprocessing
  - Full WFC3 instrument collection reprocessing for latest CALWF3 updates

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



How is JWST similar/different?

- Similar to HST
  - Same basic initial process/reprocess model
  - Same HTCondor/OWL code (shared between missions)
- Differences from HST
  - Bigger data volume – ~ 60Gb / day science data
  - File Sizes – some individual data products could approach 10Gb
  - **High-priority requirement for downlinked wavefront-sensing data
    - Used to determine JWST mirror alignment, elevated priority processing
    - We needed to learn how to manage workflow priorities under HTCondor
  - Engineering data
    - Separate data stream, loaded into databases, accessed via web service calls during science data processing for key parameters
  - More flexible "association" model (group of exposures processed as a set)
    - Dithers, mosaics, super-mosaics
    - Contemporaneous calibration exposures

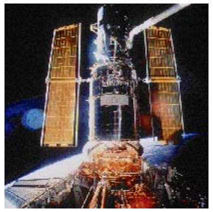


Where does JWST Data Processing stand?

- We are in the middle of JWST Data Management System (DMS) Build#6
  - We turn over to the Integration & Test team 6/1/2016

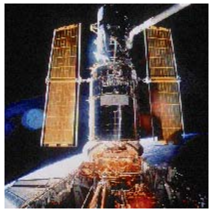
    (we probably shouldn't be here… ;)
  - 2 more planned DMS releases prior to launch in fall 2018
- Exercising the system using JWST ground test data suites
  - Science Instrument Characterization set (SIC), 1200 datasets
  - Day In the Life set (DIL), 1500 datasets
- Gaining experience with the hardware (different from HST, next slide)
- Developing and debugging our workflows
- Struggling with getting end-to-end supporting data for the test suites
- Experimenting with HTCondor daemon allocation, setting memory thresholds in .job files for certain calibration steps, reallocating memory to different core subsets, etc. ("tuning")

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



Hardware – JWST Integration & Test (pseudo-Ops)

- Currently 6 virtual machines (Vmware ESX) for daily processing / reprocessing
  - 1 dedicated for HTCondor SCHEDD, MASTER, OWLD
    - 4 cores, each w/8Gb memory
  - 1 dedicated for HTCondor NEGOTIATOR, COLLECTOR, MASTER
    - 4 cores, each w/2Gb memory
  - 4 worker machines for HTCondor STARTD, MASTER, OWLD
    - 64 cores, each w/8-12 Gb memory
  - Red Hat Enterprise Linux 6
  - Pathfinder for eventual Ops pool (which will have core counts at least comparable to the HST pool)
- Shared disk access over 10Gb network to Isilon Network Attached Storage (NAS)
- MS SQL Server databases on a clustered fail-over set of Win2012 and Win2008R2 boxes
- 1Gb network to other systems
- CycleServer (www.cyclecomputing.com) for pool monitoring and management



condor_config.local settings we tweaked

Our dominant paradigm (reprocessing) is running thousands of workflows, each of which normally doesn't take very long, on average (from 1-5 minutes), so Francesco tried to tune some HTCondor configuration settings to enable us to start our workflows more quickly.

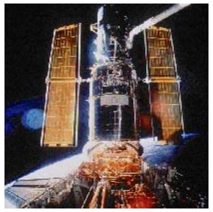
- NEGOTIATOR_INTERVAL = 1
- NEGOTIATOR_CYCLE_DELAY = 0
- NEGOTIATOR_UPDATE_AFTER_CYCLE = True
- #
- DAGMAN_SUBMIT_DELAY = 0
- DAGMAN_USER_LOG_SCAN_INTERVAL = 1
- DAGMAN_ABORT_DUPLICATES = False

# Job Hooks

- STARTER_INITIAL_UPDATE_INTERVAL = 1
- STARTER_ENVIRONMENT = "DRMAA_LIBRARY_PATH=/usr/lib64/condor/libdrmaa.so"
- STARTD_ENVIRONMENT = "DRMAA_LIBRARY_PATH=/usr/lib64/condor/libdrmaa.so"
- OWL_HOOK_PREPARE_JOB = $(OWL_RELEASE_DIR)/bin/owl_job_hook.py
- OWL_HOOK_JOB_EXIT = $(OWL_RELEASE_DIR)/bin/owl_job_hook.py
- OWL_HOOK_UPDATE_JOB_INFO = $(OWL_RELEASE_DIR)/bin/owl_job_hook.py

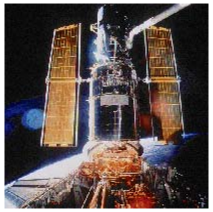
- LOG_ON_NFS_IS_ERROR = False
- DAGMAN_LOG_ON_NFS_IS_ERROR = False

STScI | SPACE TELESCOPE SCIENCE INSTITUTE



What are the challenges?

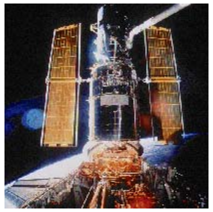
- We are still HTCondor "newbies", learning more all the time, but are <u>certain</u> that we are not yet configuring and managing the system in an optimal way (we seem to be "good enough" so far…)
  - What is the "best" allocation of HTCondor daemons across the pool?
  - What resources (memory/cpus) are necessary for the different daemons to perform well?
  - Where are our bottlenecks?
  - We learned MUCH from our first days using CycleServer to view JWST test processing in real-time
- We have history with the OPUS system that clouds our thinking
  - Move AWAY from fine-granularity workflows to "super-jobs" (process multiple observations or a full reprocessing run in a single job)
  - This approach will help us drive down the overheads we pay for each workflow that we start-up and run



What are the challenges? (cont.)

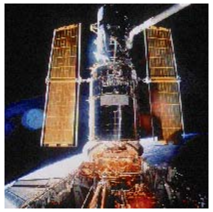
- "Train wreck" scenarios during data processing can create an operational burden to clean up and rescue failed workflows (e.g. a "hiccup" in database connectivity can derail many of our workflows)
  - Large reprocessing failures are easier to handle; just re-submit unfinished datasets back to DPQ; reprocessing can always start at the beginning
    - All inputs for data being reprocessed are already available in the archive
  - First-run processing is different
    - completeness metrics are collected so we know all expected data from the observatory was received and processed, and starting from the beginning on failure can make metrics collection more difficult
    - Grouping related exposures into combined products must wait for all pieces to arrive on the ground, so there is data accounting that occurs in the first-run that complicates failure recovery (and is unnecessary during reprocessing)

What are the challenges? (cont.)



- The population of the database table used to track dataset progress through workflows is not completely robust
  - We see sporadic HTCondor job hook failures, likely due to database connectivity issues, so Operators can have difficulty in discerning real failures from errant reports in the database (most-often the workflows completed OK, but the DB says otherwise)
  - We may back-off from real-time database insert/update in every job hook call, and instead queue DB requests for more reliable, but less real-time, capture of processing state

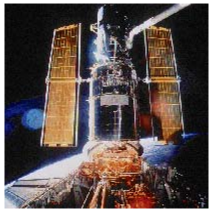
```
806592.000:  Request is held.
Hold reason: Error from
slot41@dmsops1.stsci.edu: failed to execute
HOOK_PREPARE_JOB (/usr/stsci/pipeline/owl/bin/
owl_job_hook.py)
```



What are the challenges? (cont.)

- Calibration programmers are clever folks who continue to devise new computationally-expensive algorithms that make reprocessing of the archive collection a performance burden
- Our archive users would prefer if a reprocessing run did not take many months to accomplish
  - **HTCondor to the rescue!** grab all the CPUs we can get for major reprocessing runs to speed things along

STScI | SPACE TELESCOPE SCIENCE INSTITUTE

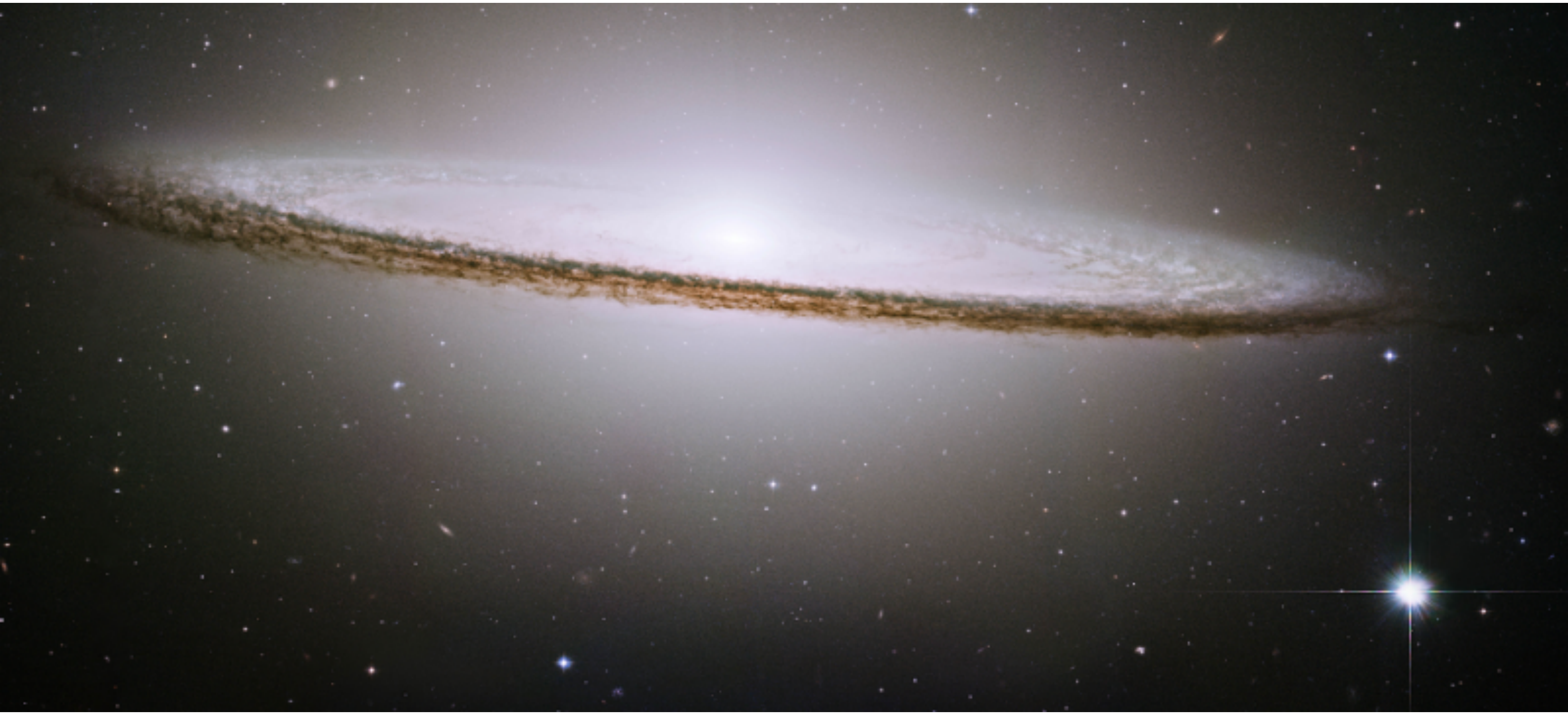


In Closing

- We feel we are on the right path for JWST Data Processing
- We are getting on more solid operational footing with HST Data Processing
- We will learn to optimize HST processing in time for JWST launch
- HTCondor was the right choice for our workflow management needs, but we did have to supplement, as most other sites likely do as well

We are VERY appreciative to the HTCondor staff, and our consultant Scott Koranda, for helpful responses to our queries and questions as we continue to spin-up on this powerful, complex system ("…I KNOW there must be a knob for that… ")

Thanks for listening!



SPACE TELESCOPE SCIENCE INSTITUTE