

# Feeding People’s Curiosity: Leveraging the Cloud for Automatic Dissemination of Mars Images

David Knight, Mark Powell  
Jet Propulsion Laboratory, California Institute of Technology  
4800 Oak Grove Dr.  
Pasadena, CA 91109  
{ 818-354-5460, 818-653-8012 }  
{ david.s.knight@jpl.nasa.gov, mark.w.powell@jpl.nasa.gov }

*Abstract*—Smartphones and tablets have made wireless computing ubiquitous, and users expect instant, on-demand access to information. The Mars Science Laboratory (MSL) operations software suite, MSL InterfaCE (MSLICE), employs a different back-end image processing architecture compared to that of the Mars Exploration Rovers (MER) in order to better satisfy modern consumer-driven usage patterns and to offer greater server-side flexibility. Cloud services are a centerpiece of the server-side architecture that allows new image data to be delivered automatically to both scientists using MSLICE and the general public through the MSL website (<http://mars.jpl.nasa.gov/msl/>).

At the core of the image processing architecture are Amazon Simple Workflow (SWF), Simple Storage Service (S3), and SimpleDB that respectively provide cloud-based workflow orchestration, object storage, and metadata storage. As a general rule, the cloud is used as a neutral ground where non-sensitive data becomes available to worker machines performing processing and ultimately the users that view the data. From an image processing perspective, early cloud availability of image data means that cloud machines can be used for the bulk of image processing tasks and elastically provisioned depending on need. From a data availability perspective, S3 is a durable, highly available storage service that does not depend on the availability of JPL’s internal network.

The overall pipeline architecture is comprised of two separate pipelines that operate in serial. The first pipeline processes images for ingestion into MSLICE and requires certain components to run on-lab at JPL. The second pipeline prepares images from multiple sources for availability to the MSL website. This second pipeline runs completely in the cloud and does not directly depend on the availability of JPL internal resources. This processing architecture allows greater flexibility for how and where MSL data is processed and puts less of a burden on the uptime of JPL’s internal resources. Total processing capacity can be elastically increased or decreased by provisioning workers on the cloud, and processes that do need to be run at JPL can be run on smaller capacity machines. The end result of this automation for the general public is that many image types appear automatically on the website in near real-time only minutes after they become available to scientists.

## TABLE OF CONTENTS

1. INTRODUCTION .....	1
2. BACKGROUND .....	1
3. TECHNOLOGICAL ANCESTRY .....	3
4. MSLICE DATA INGESTION PIPELINE .....	4

5. PUBLIC OUTREACH DISTRIBUTION PIPELINE .....	7
6. LESSONS LEARNED .....	9
7. CONCLUSION .....	9
8. ACKNOWLEDGEMENTS .....	9
REFERENCES .....	10
BIOGRAPHIES .....	10

## 1. INTRODUCTION

One of the design goals of the Mars Science Laboratory (MSL) mission was to include a multitude of instrumentation onboard the Curiosity rover. However, instrumentation on a remotely operated spacecraft involves more than the direct integration with onboard flight systems. Instrument data needs to make its way back to the eyes of the spacecraft’s scientists and engineers on Earth, and high-profile missions like MSL need to deliver enticing content to the general public, as well.

Coinciding with this most recent Mars mission is the introduction of more advanced consumer-centric technologies ranging from smartphones to social networks. Consequently, users expect new data to become available on-demand. In the context of a deep space mission, scientists, engineers, and the general public expect new spacecraft data to be made available as quickly as possible after it is terrestrially received.

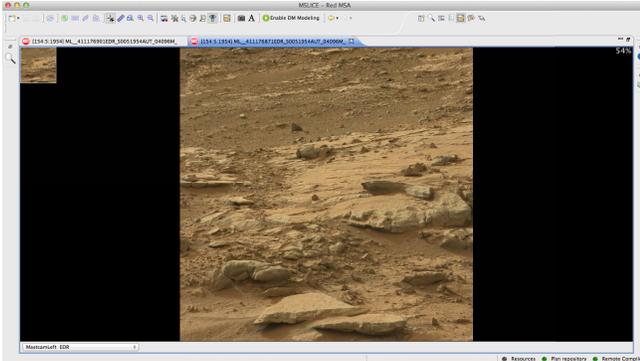
This paper describes two related data processing systems to designed to fulfill these expectations. The first system is a tactical pipeline that ingests data into the back-end system of the MSL operations software suite, enabling data consumption by operations scientists and engineers. The second system, downstream from the first system, is a public outreach distribution pipeline that makes certain types of captured Curiosity images available for public consumption through the MSL public website.

## 2. BACKGROUND

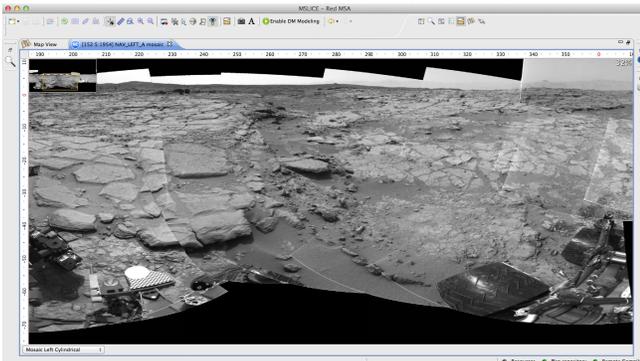
### *MSL Operations Software*

The operations software package fed by the tactical data ingestion pipeline is the MSL InterfaCE (MSLICE) suite, developed in JPL’s Ops Lab. MSLICE is designed to provide an integrated environment for performing many of

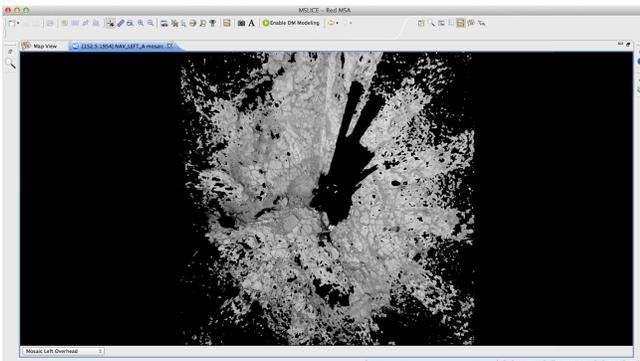
the different functions required to operate Curiosity. Major functionality includes high-level timeline planning of science and engineering activities to be executed by the rover, low-level sequencing of the actual commands to uplink to the rover, viewing of image data products downlinked from the rover, and data visualization including panoramic images, overhead maps, and 3D renderings.



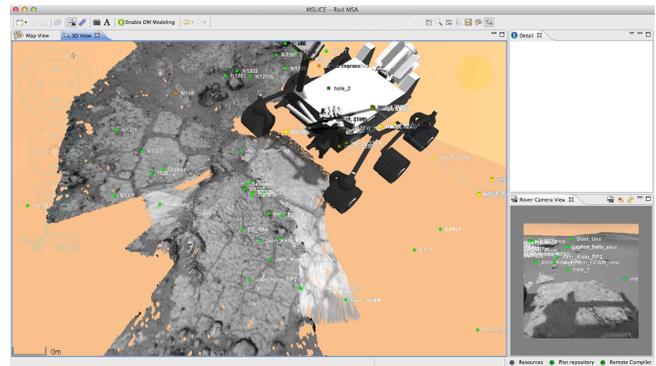
**Figure 1 – A Mastcam image being viewed in MSLICE**



**Figure 2 – A Navcam cylindrical (panoramic) mosaic being viewed in MSLICE**



**Figure 3 – A Navcam orthographic (overhead) mosaic being viewed in MSLICE**



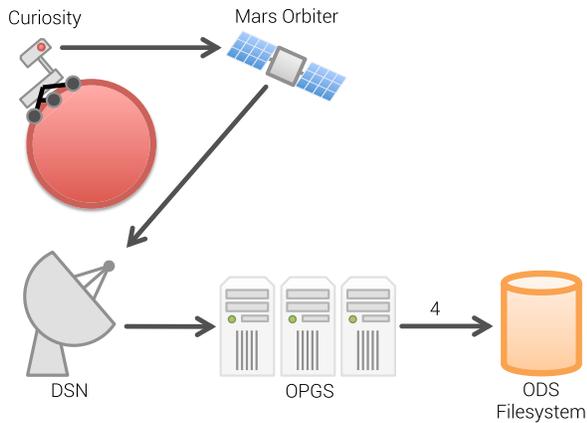
**Figure 4 – The 3D view of MSLICE showing terrain rendered Navcam range data and imagery**

In the larger context of the MSLICE system, the ingestion pipeline primarily fulfills the role of processing downlinked data products into forms that can be efficiently downloaded and viewed remotely by MSLICE users. This task involves uploading non-sensitive data products to the cloud for distribution, and also cataloging data products so that they can be discovered through search queries. Finally, the pipeline does generate some derivative data products that are specific to MSLICE, such as cylindrical (panoramic) and orthographic (overhead) mosaic images used for instrument targeting and providing general spatial awareness [1] [2]. Mosaic images are composite images composed of multiple individual images that have been transformed and stitched together.

### *Upstream Systems*

Data flows through a variety of upstream systems before it is ready to be ingested into MSLICE. Naturally, data needs to be transmitted from the Curiosity rover on Mars back to Earth, but additional processing takes place to transform sensor readings into a consumable format made available on the mission's network filesystem.

Under nominal operation, Curiosity communicates with Earth through the assistance of a Mars orbiter acting as a relay. Either the Odyssey orbiter or the Mars Reconnaissance Orbiter (MRO) can act as the relay, and the visibility of the orbiter from both Earth and the rover is the primary limiting factor in duration and frequency of communication windows. Radio signals are transmitted and received on Earth through the Deep Space Network (DSN), which is composed of a network of three ground stations located in the United States, Spain, and Australia. A primary downlink of information from Curiosity to Earth occurs nearly once per Earth day, however the time of the downlink drifts due to the longer length of a Martian day and the varying visibility of the Mars orbiter to the rover and to Earth.



**Figure 5 – High-level data flow diagram of Curiosity data transmission and upstream processing**

Once radio signals are decoded on Earth, the Operations Product Generation Subsystem (OPGS) extracts information from low-level spacecraft data and outputs individual data product files. These files fall into two different categories: Experiment Data Record (EDR) and Reduced Data Record (RDR). An EDR file contains the original, un-modified information captured by one of Curiosity’s instrument. An RDR file contains derived information like depth data inferred from stereo image pairs.

These EDR and RDR data product files are output in a format called the Planetary Data System (PDS) format. The PDS format includes an extensive header of metadata providing information about the rover’s state when the data was acquired, such as timestamps, location information, and geometry information. OPGS writes PDS files to an MSL-specific network filesystem called the Operational Data Store (ODS) that can be accessed by various downstream applications, including the MSLICE data ingestion pipeline.

### 3. TECHNOLOGICAL ANCESTRY

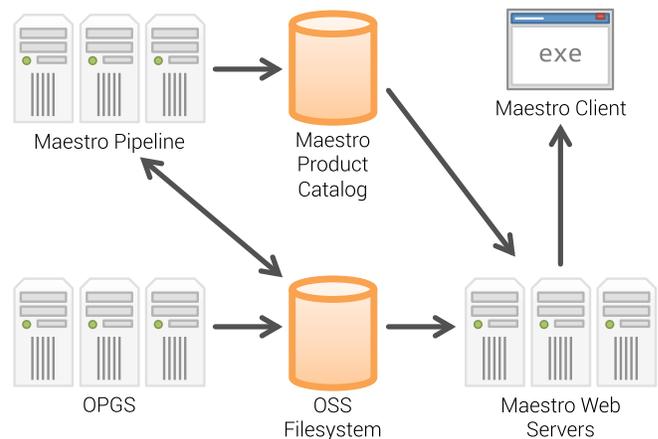
#### *Mars Exploration Rovers*

The design of many components of MSLICE, including the ingestion pipeline, grew out of an operations application called Maestro developed for the Mars Exploration Rovers (MER) mission. Maestro’s scope is narrower than that of MSLICE in that it lacks sequencing functionality and some of the more advanced planning functions, but it includes similar image product viewing functionality as well as its own data ingestion pipeline. Similar to how data is transmitted and processed on the MSL mission, the Maestro ingestion pipeline is downstream from OPGS and accesses data products from a MER-specific network filesystem called the Operation Storage Service (OSS).

The core component of Maestro’s ingestion pipeline is a server-side Perl program called the dp-daemon [3]. PDS image files on the OSS filesystem are organized by sol (the count of Martian days since spacecraft landing) and by instrument, and the dp-daemon uses this structure along

with a heuristic to routinely poll for recently added image products.

Upon discovery, the dp-daemon catalogs new image products in a MySQL database using a subset of the metadata fields found within PDS header. The catalog enables users to issue basic searches for subsets of images based on fields like sol or instrument. After cataloging, the dp-daemon writes a compressed copy of the PDS image to the OSS filesystem using either JPEG or PNG compression depending on whether the image product type necessitates lossy or lossless compression. These compressed image copies are then served to Maestro clients using an Apache web server.



**Figure 6 – Architecture and data flow diagram of Maestro data ingestion**

#### *Distributed Data Processing with Polyphony*

Polyphony is the name of a distributed data processing system prototyped in the JPL Ops Lab [2]. The system is built on top of various cloud services: Amazon Simple Queuing Service (SQS), Amazon Simple Storage Service (S3), and Amazon SimpleDB. The core concept of Polyphony is that the worker computers that perform the actual data processing are kept as generic as possible so that additional worker computers can be added and removed from the system with ease.

Amazon SQS is used to maintain a cloud-based job queue that contains descriptions of data processing jobs that need to be performed [2]. It is a requirement that the job descriptions be self-contained, meaning that a description contains all necessary information needed for a worker computer to execute the job in a stateless environment. An analogy of this restriction in Java is writing a static method that does not make use of global variables, resulting in a pure function. This requirement is what keeps the worker computers generic because jobs are essentially performed in a stateless environment and can run on any worker computer at any time.

What makes a computer a worker computer is simply the running of a processing daemon that contains the software

necessary to execute different kinds of job descriptions. This daemon is responsible for repeatedly downloading a new job description from the job queue and performing the actual data processing outlined in the job description.

In order to start data processing, a user or upstream process simply adds job descriptions to the Amazon SQS job queue. Assuming a non-zero number of worker computers are running, data processing starts automatically. Worker computers can be added or removed dynamically depending on anticipated load [2].

ASDFASFASDFASDFKSAJFJSKJDF !!!!

#### 4. MSLICE DATA INGESTION PIPELINE

##### Architecture

The MSLICE data ingestion pipeline design is an evolution of the distributed, queue-based Polyphony system described in the Technological Ancestry section. The system still operates based on the concept of having a queue of jobs that need to be performed, and a pool of worker computers running a processing daemon that receives jobs from the queue and subsequently performs them. However, this pipeline does not use Amazon SQS to manage the job queue, but instead uses the newer Amazon Simple Workflow (SWF) service. Amazon SQS is a fairly primitive service allowing for messages to be stored in a queue. In comparison, Amazon SWF is a high-level service based around task lists, which are queues of workflow executions.

A workflow, referred to by a workflow type and version, is a definition of a series of processes to be executed on a worker computer, akin to a pipeline. The sub-units of processing that a workflow is comprised of, like pipeline stages, are called activities and similarly are referred to by an activity type and version. It follows that a particular instance of a workflow being executed on a worker computer is called a workflow execution. A workflow execution specifically includes a set of input parameters to the workflow.

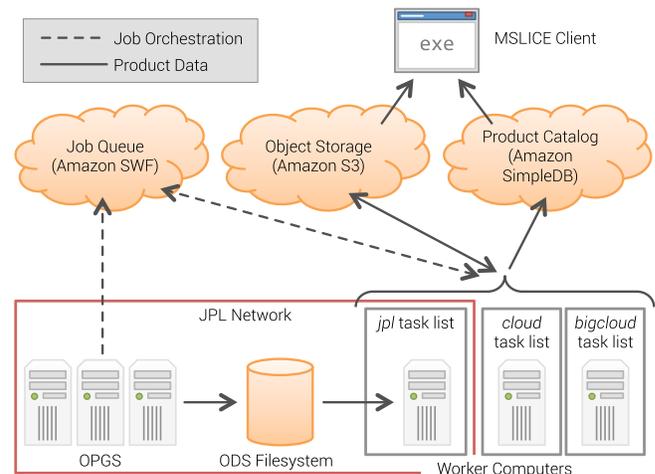
The MSLICE ingestion pipeline is setup with three different Amazon SWF task lists: *jpl*, *cloud*, and *bigcloud*. The *jpl* task list queues workflow executions that need to be run on worker computers with direct access to the ODS filesystem, which is a JPL internal network asset. The *cloud* task list queues workflow executions that do not require JPL internal network access and can consequently be run on worker computers provisioned from the cloud. The *bigcloud* task list queues workflow executions that do not require JPL internal network access and also require the worker computer to be equipped with additional RAM. The data processing daemon running on the worker computers needs to be configured to listen to the correct task list based on the worker computers' available resources. This configuration

prevents a workflow execution from being received by a worker computer that is unable to complete its assignment.

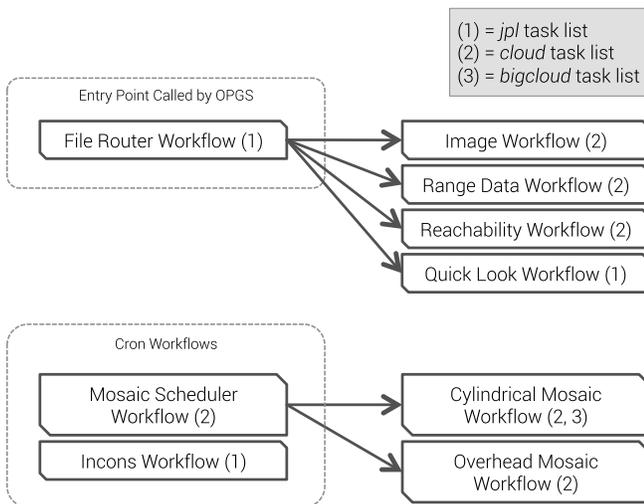
In a broader sense, this scheme of segmenting task lists allows for increased flexibility when provisioning computers that will be used as workers. Since a majority of the processing load occurs in the *cloud* and *bigcloud* task lists, computers can be easily provisioned in the cloud in anticipation of high processing load events.

As outlined in the Background section, this pipeline ingests data products produced by OPGS by reading them from the shared ODS filesystem. However, unlike the Maestro data ingestion pipeline used with MER, the MSLICE data ingestion pipeline does not discover new data products through polling. OPGS is configured to submit a new File Router Workflow execution to the *jpl* task list in Amazon SWF for every data product that gets produced. These workflow executions include the path on the ODS filesystem of the new data product as an input parameter.

The File Router Workflow type is designed to be the single entry-point to the MSLICE data ingestion pipeline. Assuming that the input data product is of a type that needs to be ingested, the File Router Workflow's purpose is to upload the incoming data product file from the ODS filesystem to Amazon S3 and to subsequently launch the correct type of workflow execution in order to process the type of data product being submitted. The uploading of data product files to Amazon S3 early in the flow of the pipeline enables significant use of the *cloud* and *bigcloud* task lists and loosens the restriction on how many worker computers need to have access to the JPL internal network.



**Figure 7 – Architecture diagram depicting major functional components of the MSLICE data ingestion pipeline**



**Figure 8 – Workflow diagram depicting the call hierarchy of different workflow types**

In general, there is a different workflow type responsible for processing each of the major data product types. For historical reasons, the actual names of these workflow types in source code do not correlate exactly with the names of the data product types themselves. For clarity in this paper, the workflow type names have been changed to better match their corresponding data product types.

*Distributed Cron Workflows*—The majority of the workflow types that make up the data ingestion pipeline sequentially perform a finite set of tasks before completing. However, some workflow types are designed to run indefinitely. These workflow types, dubbed cron workflows, are used in the pipeline to implement polling mechanisms. A cron workflow runs an activity repeatedly on a fixed interval similar to a cron job on a UNIX operating system. Unlike a local cron job, cron workflows running through Amazon SWF are distributed and can run on any worker computer polling the proper task list. In other words, distributed cron workflows are not specific to any one machine and are consequently immune from single machine failure provided other worker computers are available.

The data ingestion pipeline uses cron workflows to accomplish a couple different miscellaneous tasks. The Mosaic Scheduler Workflow triggers the regeneration of mosaic images for locations where the mosaic imagery is out of date (see MSLICE Data Ingestion Pipeline; Data Product Types; Mosaics). The Incons Workflow is tasked with polling for the existence of a type of arm kinematics file not produced by OPGS. Upon discovery, the Incons Workflow uploads copies of new kinematics files to Amazon S3 after they become available on the ODS filesystem.

#### Data Product Types

A variety of different data product types need to be ingested in order to support the spatial and imaging functions of MSLICE that enable rover operators to know the rover’s

location, the location of surrounding objects, and the general physical context of Curiosity’s surroundings. The different types of data products ingested from OPGS are differentiated with a three-character code in the filename. A table of the different product type codes as well as functional descriptions of the different product types follows.

**Table 1. Data product type codes corresponding to different product categories ingested from OPGS**

Data Product Categories	Data Product Type Codes
<b>Images</b>	EDR, ERD, LIN, RAD, RAS
<b>Range Data</b>	RNG, UVW
<b>Reachability Map</b>	ARM
<b>Mosaics</b>	N/A (non-OPGS)
<b>QuickLooks</b>	N/A (non-OPGS)

*Images*—Curiosity is equipped with a variety of different cameras: two stereo pairs of hazard avoidance cameras (Hazcams), one stereo pair of navigational cameras (Navcam), a chemical camera (Chemcam), two mast cameras (Mastcam), the Mars Hand Lens Imager (MAHLI), and the Mars Descent Imager (MARDI). Consequently, images are the most foundational type of data product imported into MSLICE. For every image captured by Curiosity, both a raw version and a radiance-corrected version are produced by OPGS and ingested. MSLICE allows operations scientists and engineers to perform basic viewing functions like zooming, panning, and marking points of interest that are shared amongst users. Because image products contain metadata about the rover’s location, orientation, pose, and camera geometry, points of interest created in the coordinate system of one image are transformed into the coordinate systems of nearby images taken at the same location. These transformations allow points of interest to be viewable

Three primary activities occur in the Image Workflow that respectively perform format conversion, image tiling, and cataloging of image products. The first activity converts the PDS-formatted image into JPEG and PNG formatted files that are uploaded to Amazon S3. The metadata in the headers of the PDS file are also extracted and uploaded to the cloud as a human-readable text file with an LBL extension. These image files are available on the cloud for MSLICE users that wish to download a copy of an image product to disk in a common file format.

The second activity uploads a tiled image pyramid representation of the image product to Amazon S3. A tiled image pyramid is a data structure that allows for responsive viewing of large images over a network. The source image is resized three times using a scaling factor of ½. At each resolution, the image is broken up into tiles of size 256 by 256 pixels that get encoded as JPEG2000 files. These tile files are the data that gets downloaded when users are viewing image products in MSLICE. This data structure allows MSLICE to progressively load and display only the

parts (and detail level) of the image that a user is currently viewing.

The third activity creates an entry for this image product within a product catalog table in Amazon SimpleDB. If the image product is from either the Navcams or Mastcams, an entry also gets created in an outdated mosaic table in SimpleDB. This database table stores a list of mosaics that need have become out of date and need to be regenerated (see MSLICE Data Ingestion Pipeline; Data Product Types; Mosaics).

Once the three primary activities are complete, an optional fourth activity can trigger the automatic public release of the image product. This signal gets sent for Hazcam, Navcam, and Chemcam EDR image products, in which case a new workflow execution is submitted to the public outreach distribution pipeline (see Public Outreach Distribution Pipeline).

*Range Data*—Since MSL is equipped with multiple stereo camera pairs (see MSLICE Data Ingestion Pipeline; Data Product Types; Images), OPGS infers range and normal information for image products where a corresponding left-right pairing exists. Range data is used in MSLICE to allow users to select a point in an image product and receive an estimate of the depth of that imaged surface from the camera. The availability of this depth information is vital for operations planning of many of the science instruments simply because it informs operators of how far objects are from the rover.

The Range Data Workflow uses the range and normal data products to produce a series of 32 by 32 pixel tiles that hold a 4-dimensional single-precision floating-point vector for each pixel of the corresponding image product. This vector represents the range of the surface normal from the camera origin and the Cartesian direction of the surface normal. Additionally, a JPEG2000 formatted range map is produced. The tile and range map files are all uploaded to Amazon S3 after creation.

*Reachability Data*—Since many of Curiosity’s instruments are mounted on the arm, operators need the ability to estimate the extent of the rover’s surroundings that are within reach of the arm. OPGS specifically produces an arm reachability data product for whenever range data is available from a camera stereo pair. MSLICE uses this information to optionally draw an overlay on top of image products that indicates what regions are within reach of Curiosity’s arm.

The reachability maps provide reachability ratings for eight different arm poses for every pixel of an image product. The reachability rating is a 2-bit value where 0 means completely unreachable. The set of ratings for an image pixel is encoded as a 16-bit integer value, and the Reachability Workflow encapsulates this data as a 16-bit grayscale PNG image file that gets uploaded to Amazon S3.

*Mosaics*—Unlike images, range data, and reachability data, mosaics are a derivative product that is generated by the MSLICE data ingestion pipeline itself. A mosaic is a larger image that is produced by stitching together many individual images. The pipeline automatically produces cylindrical (panoramic) and orthographic (overhead) perspective mosaics from Navcam images taken from a single location, and the pipeline can also produce similar cylindrical mosaics from Mastcam imagery. Due to the resulting high-resolution of Mastcam mosaic images, the workflow executions to produce these products specifically make use of the *bigcloud* task list. However, Navcam mosaics are produced using workflow executions on the regular *cloud* task list.

MSLICE presents mosaics with fluid zooming and panning functionality in order to give users an unfettered view of the rover’s surroundings. Additionally, points of interest from individual image products are also displayed in the cylindrical mosaic viewer. This visualization functionality gives users a better spatial context of the rover’s environment and also provides a central location for scientists and engineers to refer to and discuss points of interest.

Generation of mosaics is enabled by the fact that image products contain a wealth of spatial metadata that allows for an accurate 3D coordinate system to be established. Unlike purely visual mosaicking methods that estimate geometric transforms based on common key points in images, the data ingestion pipeline uses the provided camera geometry metadata to project all relevant images into a coordinate system that provides accurate pointing and targeting relative to the rover.

As mentioned previously, an Amazon SimpleDB database table contains entries referring to outdated mosaics that require regeneration (see MSLICE Data Ingestion Pipeline; Data Product Types; Images). An entry is added to this table whenever a new Navcam or Mastcam image product is processed. A cron workflow called the Mosaic Scheduler Workflow is responsible for reading this database table and asynchronously submitting new workflow executions of Cylindrical Mosaic Workflow and/or Overhead Mosaic Workflow to Amazon SWF.

Because of the high resolution of cylindrical mosaics, the Cylindrical Mosaic Workflow generates a mosaic in stripes that are 256 pixels tall and as wide as the final image. The processing of these stripes is performed using multiple activities in parallel. After stripes have been created, another phase of parallel activities repeatedly tiles and down samples the individual mosaic stripes into a JPEG2000 tiled image pyramid representation that is uploaded to Amazon S3, enabling responsive viewing in MSLICE (see MSLICE Data Ingestion Pipeline; Data Product Types; Images).

The resolution of the orthographic mosaics does not necessitate generating the mosaic image piecemeal using stripes. Consequently, the Orthographic Mosaic Workflow

generates the mosaic in a single activity and creates a JPEG2000 tiled image pyramid of the mosaic that gets uploaded to Amazon S3 in a subsequent activity.

Similar to individual image products, the creation of new Navcam mosaics causes a new workflow execution to be sent to the public outreach distribution pipeline. This workflow execution is responsible for triggering a public release of the new mosaic image (see Public Outreach Distribution Pipeline).

**QuickLooks**—MSLICE allows scientists and engineers to upload documents to be cataloged and shared with other users. These files are stored as ZIP archives on the ODS filesystem, and a new File Router Workflow execution is sent to Amazon SWF upon uploading. The Quick Look Workflow is responsible for uploading the document to Amazon S3 and cataloging the document so that it can be discovered through MSLICE search. Additionally, if the document is a PDF file, a PNG thumbnail of the first page of the document is generated and uploaded to Amazon S3.

*Implementation*

While the architecture of the MSLICE data ingestion pipeline would allow for most of the worker computers to run in the cloud, most of the infrastructure still runs at JPL under nominal operations. The primary worker computer is a virtualized server equipped with 128 GB of RAM and two Intel Xeon X5570 CPUs. This machine is used to run the processing daemons that listen to both the *jpl* and *cloud* task lists.

At the time of this writing, worker computers for the *bigcloud* task list have not yet been deployed for production. In the future, it is likely that these worker computers would be deployed on Amazon Elastic Cloud Compute (EC2) using m1.large sized instances, which are equipped with 7.5 GiB of RAM and two dual-core virtualized processors [4].

The data processing daemon itself is written in Java 6 with the Open Services Gateway Initiative (OSGi) framework. The OSGi framework allows the daemon to be written as a series of plug-in modules, many of which contain code shared with the MSLICE client. This plug-in architecture minimizes code duplication and improves code maintainability. Interaction with various cloud services is accomplished through the Amazon Web Services Software Development Kit (AWS SDK) for Java.

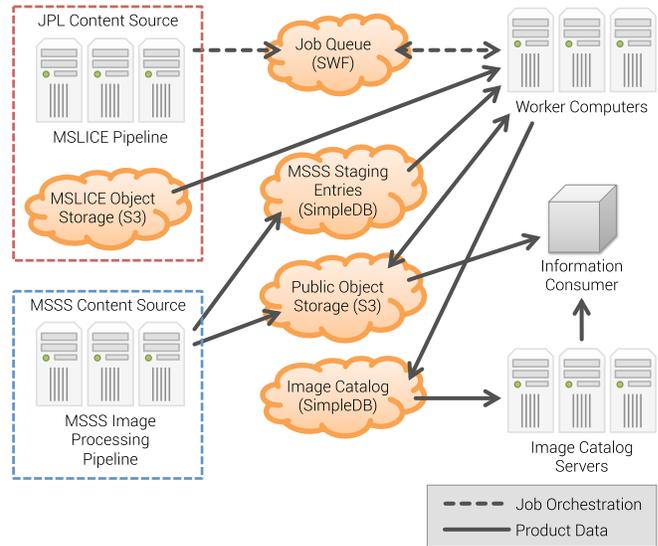
**5. PUBLIC OUTREACH DISTRIBUTION PIPELINE**

*Architecture*

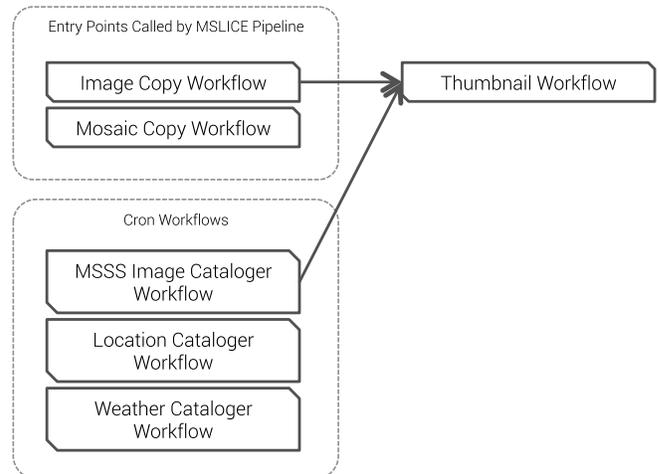
The public outreach distribution pipeline is designed to release generally unmodified images taken by Curiosity to the general public in a timely and automatic fashion. It's based on the same technology stack of cloud services as the MSLICE data ingestion pipeline: Amazon SWF, Amazon S3, and Amazon SimpleDB. However, this pipeline runs

under a completely separate Amazon Web Services account for both isolation and organizational purposes.

This pipeline operates entirely in the cloud because it does not depend on any JPL-internal network resources. Image input comes from two different sources: OPGS image products processed at JPL and images processed by Malin Space Science Systems (MSSS).



**Figure 9 – Architecture diagram depicting major functional components of the public outreach distribution pipeline**



**Figure 10 – Workflow diagram depicting the call hierarchy of different workflow types**

The high-level technical objective of the distribution pipeline is to upload either JPEG or PNG formatted images to cloud storage, upload image thumbnails to cloud storage, and to keep a catalog of available images. Amazon S3 is used for cloud storage due to being both durable and capable of directly serving large amounts of data. Amazon SimpleDB tables are used to store the catalog of available images, and a simple web server allows other applications to

perform HTTP queries against the catalog. Similar to the Maestro and MSLICE designs, the outreach image catalog allows other applications to quickly discover what image products exist in the system.

The distribution pipeline produces four different thumbnail sizes, which are produced by a Thumbnail Workflow. The resizing policy is based on width and height constraints, and source images are downsampled such that the aspect ratio is preserved and all dimensional constraints are met. When both width and height constraints are present, for example, the source image is essentially shrunk to fit inside a bounding box. Source images that already meet all constraints are not resized, disallowing the enlarging of source images. Thumbnail image files inherit the format of the source image (either JPEG or PNG) and are uploaded to Amazon S3 using a programmatic filename suffix. The different thumbnail sizes are shown below.

**Table 2. Thumbnail size specifications for raw MSL images released to the public**

filename suffix	width constraint	height constraint
“-br”	500 px	none
“-br2”	1024 px	none
“-thm”	160 px	120 px
“-thm2”	320 px	240 px

Finally, similar to the MSLICE data ingestion pipeline, cron workflows are used to accomplish various tasks. The MSSS Image Cataloger Workflow periodically checks if new images submitted by MSSS need to be cataloged (see Public Outreach Distribution Pipeline; MSSS Originating Images). The Location Cataloger Workflow routinely updates a static file on Amazon S3 with a list of rover locations. The Weather Cataloger Workflow periodically uploads to Amazon S3 static files containing Martian surface conditions.

#### *OPGS Originating Images*

As indicated previously, certain image products processed in the MSLICE data ingestion pipeline are sent to the outreach distribution pipeline (see MSLICE data ingestion pipeline; Data Product Types). Specifically, the outreach distribution pipeline receives EDR image products taken by the Navcams, Hazcams, or Chemcam, and it also receives Navcam mosaic images generated upstream.

For images, the upstream processing of these specific types of image products triggers the creation of an Image Copy Workflow execution in the outreach distribution pipeline. The first activity of this workflow is responsible for both writing a publicly accessible JPEG file of an image product and creating a corresponding entry in the image catalog. The JPEG file is copied from an MSLICE-specific bucket of Amazon S3 to a publicly accessible outreach bucket. The catalog entry is created using PDS metadata loaded from the image product’s corresponding LBL file located in the MSLICE-specific Amazon S3 bucket, and the catalog entry

is written to an image catalog database table in Amazon SimpleDB.

The second activity of the Image Copy Workflow involves launching a child workflow execution of type Thumbnail Workflow to produce thumbnails based on the input image product.

Incoming mosaic images are processed similarly, except that thumbnails do not get generated. Either a JPEG or PNG copy of the image file is copied from the originating MSLICE-specific bucket to a publically accessible bucket on Amazon S3. The corresponding metadata from the mosaic image is used to create an entry in a mosaic catalog database table in Amazon SimpleDB.

#### *MSSS Originating Images*

Images captured from Curiosity’s Mastcams, MAHLI, and MARDI cameras are processed by MSSS before being submitted to the outreach distribution pipeline. Even though the image products come from a different upstream source, the outreach distribution pipeline performs the same thumbnailing and cataloging operations, allowing for MSSS image products to be accessed in the same manner as image products generated with OPGS.

Because the MSSS image processing pipeline runs at a different institution, ingestion of its output image products is less tightly coupled with this pipeline compared to the coupling between this pipeline and the MSLICE data ingestion pipeline. When new image products are output at MSSS, their image processing pipeline invokes a submission script that directly uploads a JPEG file to the publicly accessible outreach bucket on Amazon S3 and makes an entry in a staging domain of Amazon SimpleDB.

A cron workflow in the outreach distribution pipeline polls the staging database table in Amazon SimpleDB every five minutes. When new entries are present in the staging database table, a workflow execution of the Thumbnail Workflow is run for each new MSSS image product, and corresponding catalog entries are written to the image catalog database table.

#### *Implementation*

Since access to JPL’s internal network resources is not required, all infrastructure of the outreach distribution pipeline runs in the cloud. Individual worker computers and image catalog servers are provisioned on Amazon EC2. Publicly accessible files are stored on Amazon S3, the image catalog is stored in Amazon SimpleDB, and Amazon SWF is used to dispatch tasks to worker computers.

Two m1.medium sized EC2 instances running Amazon Linux 2012.09 act as worker computers for the pipeline. These virtualized machines are equipped with 3.75 GiB of RAM and a single 64-bit dual-core CPU [4]. The processing daemon running on these machines is written in Java 6 with the OSGi framework and the AWS SDK for Java.

There are currently two image catalog servers running on m1.small EC2 instances. These virtualized machines are equipped with 1.7 GiB of RAM and a single single-core 64-bit CPU [4]. These image catalog servers currently run Amazon Linux 2012.09. Both of these machines run behind an Elastic Load Balancer (ELB), which is the point of access for downstream systems. The load balancer allows for additional image catalog servers to be added and removed dynamically based on the anticipated load from downstream systems. The server software is written in Java 6 with the OSGi framework, AWS SDK for Java, and Restlet framework.

The image catalog servers accept queries by Martian sol and optionally allows for filtering by instrument. Queries are executed against Amazon SimpleDB, and results are delivered back to clients as XML documents. In particular, this XML data is ingested by JPL's Mars Science Laboratory website servers and used to populate the Raw Image Gallery, enabling the automatic release of images taken by Curiosity to the general public.

## 6. LESSONS LEARNED

The path to implementing any moderately sized system involves learning lessons along the way, and the development of both the MSLICE data ingestion pipeline and the public outreach distribution pipeline were no exception.

First, while the use of a queue and workflow service, like Amazon SWF, simplifies many synchronization issues related to when and where code is executed, it does not solve all synchronization issues. Workflows and activities should be written to depend only on the input to the workflow or activity, similar to static methods in Java. Use of global state in the processing daemons can lead to unintended race conditions and should be used carefully and only when necessary to avoid inconsistent states across worker computers that should be identical.

Second, while verbose, it was found to be a good pattern to separate cron workflows from the payload that they are intended to execute. In other words, a cron workflow should only contain enough logic to launch a separate workflow on a specific interval. The separate workflow should contain the logic for the actual tasks that need to be executed. This separation isolates the cron logic from failures in the payload logic and can be thought of as forking a process in UNIX.

Third, adjusting Amazon SWF's timeout and retry policies is an art and not a science. Short timeouts make workflows and activities brittle in the face of variables like network latency, and long timeouts impose excessive delays before activities or workflows are retried.

Finally, based on usage patterns it actually seems like the image catalog servers in the public outreach distribution

pipeline are unnecessary. Queries to the image catalog servers only involve a single Martian sol at a time. Consequently, the image catalog information could be organized in a series of static XML files with systematic filenames and a root manifest of what catalog files are available and when they were last updated. Because the image catalog would be read as a collection of static files it would be vastly more scalable, with the hosting load abstracted behind a storage service like Amazon S3. This concept is a desirable future improvement that should be implemented.

## 7. CONCLUSION

Both of the pipelines described in this paper are built upon the same basic cloud services. Use of these cloud services abstracts away the complexity of major functional blocks of the pipeline systems and also provides greater flexibility for deployment. In particular, the use of Amazon SWF allows the pool of worker computers to be grown and shrunk as need sees fit. And in cases where organizational internal network access is not required, virtually all system infrastructure can be run in the cloud.

Taking a broader view, the MSLICE pipeline accomplishes its goal of automatically processing and making new image products available to MSLICE users. Additionally, many of these same images are made available to the general public directly after they become available to mission scientists and engineers.

## 8. ACKNOWLEDGEMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## REFERENCES

- [1] M. Powell. "A Scalable Image Processing Framework for Gigapixel Mars and Other Celestial Body Images" IEEE Aerospace Conference, Big Sky, Montana, March 6-13, 2010.
- [2] K. Shams, M. Powell, T. Crockett, J. Norris, R. Rossi, T. Soderstrom. "Polyphony: A Workflow Orchestration Framework for Cloud Computing" IEEE/ACM 10th International Conference on Cluster, Cloud and Grid Computing (CCGrid), Melbourne, Australia, May 17-20, 2010.
- [3] J. Fox, J. Norris, M. Powell, K. Rabe, K. Shams. "Advances in distributed operations and mission activity planning for Mars surface exploration" AIAA 9th International Conference on Space Operations (SpaceOps), Rome, Italy, June 19-24, 2006.
- [4] "Amazon EC2 Instance Types," Amazon EC2 Details, January 14, 2013. [Online]. Available: <http://aws.amazon.com/ec2/instance-types/>

## BIOGRAPHIES



*David Knight is an Application Software Engineer at NASA's Jet Propulsion Laboratory (JPL) currently working on the Mars Science Laboratory Interface (MSLICE) team in the Ops Lab. As a college intern in 2010, he previously developed a WebGL front-end for rendering terrain data from the Lunar Mapping and Modeling Project (LMMP). He received a B.S. in Electrical Engineering from Oregon State University in 2009 and a M.S. in Electrical Engineering from Stanford University in 2011.*



*Mark Powell is a Senior Computer Scientist at the Jet Propulsion Laboratory, Pasadena, CA. Since 2001 Mark has created apps for visualization and control of robots that explore Earth, sea, sky, space, moons and planets. Mark is the product lead for many of the command and control apps for the Curiosity Mars Rover. These apps afford environment visualization, collaborative planning, health and safety monitoring, simulation, command sequence authoring and validation to the Curiosity science and engineering teams. These apps are known collectively as the Mars Science Laboratory Interface (MSLICE). Mark also supports Curiosity as an Engineering Camera (ECAM) Uplink Lead. He supported the 2004 Mars Exploration Rover (MER) mission operations as a Science Downlink Coordinator, facilitating the timely downlink and analysis of science data from the rovers. He received the NASA Software of the Year Award for his work on the Science Activity Planner science visualization and activity planning software used for MER operations. He also received the Imager of the Year award from Advanced Imaging Magazine for his work on Maestro, the publicly-available version of the Science Activity Planner for MER. Mark also supports a variety of other projects at JPL including the Aerobot aerial robotic vehicle, the Cassini mission to Saturn, and the ATHLETE prototype asteroid-hopping robotic vehicle.*