Trident: Scalable Compute Archives - Workflows, Visualization, and Analysis

Arvind Gopu[a], Soichi Hayashi[a], Michael D. Young[a], Ralf Kotulla[b], Robert Henschel[a], and Daniel Harbeck[c]

[a]Indiana University, 2709 E 10th St., Bloomington, IN 47408, USA
[b]University of Wisconsin - Madison, 475 N Charter St, Madison, WI 53706, USA
[c]WIYN, INC., 950 N. Cherry Ave., Tucson, AZ 85719, USA

ABSTRACT

The Astronomy scientific community has embraced Big Data processing challenges, e.g. associated with time-domain astronomy, and come up with a variety of novel and efficient data processing solutions. However, data processing is only a small part of the Big Data challenge. Efficient knowledge discovery and scientific advancement in the Big Data era requires new and equally efficient tools: modern user interfaces for searching, identifying and viewing data online without direct access to the data; tracking of data provenance; searching, plotting and analyzing metadata; interactive visual analysis, especially of (time-dependent) image data; and the ability to execute pipelines on supercomputing & cloud resources with minimal user overhead or expertise even to novice computing users. The Trident project at Indiana University offers a comprehensive web- and cloud-based microservice software suite that enables the straight forward deployment of highly customized Scalable Compute Archive (SCA) systems – including extensive visualization and analysis capabilities – with minimal amount of additional coding. Trident seamlessly scales up or down in terms of data volumes and computational needs, and allows feature sets within a web user interface to be quickly adapted to meet individual project requirements. Domain experts only have to provide code or business logic about handling/visualizing their domain's data products and about executing their pipelines and application workflows. Trident's microservices architecture is made up of light-weight services connected by a REST API and/or a message bus; a web interface elements are built using NodeJS, AngularJS, and HighCharts JavaScript libraries among others while backend services are written in NodeJS, PHP/Zend, and Python. The software suite currently consists of (1) a simple Workflow execution framework to integrate, deploy, and execute pipelines and applications (2) a Progress service to monitor workflows and sub-workflows (3) ImageX, an interactive image visualization service (3) an authentication & authorization service (4) a Data service that handles archival, staging and serving of data products, and (5) a Notification service that serves statistical collation and reporting needs of various projects. Several other additional components are under development. Trident is an umbrella project, that evolved from the One Degree Imager - Portal, Pipeline, and Archive (ODI-PPA) project which we had initially refactored toward (1) a powerful analysis/visualization portal for Globular Cluster System (GCS) survey data collected by IU researchers, 2) a data search and download portal for the IU Electron Microscopy Center's data (EMC-SCA), 3) a prototype archive for the Ludwig Maximilian University's Wide Field Imager. The new Trident software has been used to deploy (1) a metadata quality control and analytics portal (RADY-SCA) for DICOM formatted medical imaging data produced by the IU Radiology Center, 2) Several prototype workflows for different domains, 3) a snapshot tool within IU's Karst Desktop environment, 4) a limited component-set to serve GIS data within the IU GIS web portal. Trident SCA systems leverage supercomputing and storage resources at Indiana University but can be configured to make use of any cloud/grid resource, from local workstations/servers to (inter)national supercomputing facilities such as XSEDE.

Keywords: Microservices, Science Gateway, IU Trident, Javascript, NodeJS, AngularJS, Docker

Further author information, contact agopu@iu.edu

Software and Cyberinfrastructure for Astronomy III, edited by Gianluca Chiozzi,
Juan C. Guzman, Proc. of SPIE Vol. 9913, 99131H · © 2016 SPIE
CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2233111

1. INTRODUCTION

The Astronomy scientific community has embraced Big Data processing challenges, e.g. associated with time-domain astronomy, and come up with a variety of novel and efficient data processing solutions. However, data processing is only a small part of the Big Data challenge. Efficient knowledge discovery and scientific advancement in the Big Data era requires new and equally efficient tools: modern user interfaces for searching, identifying and viewing data online without direct access to the data; tracking of data provenance; searching, plotting and analyzing metadata; interactive visual analysis, especially of (time-dependent) image data; and the ability to execute pipelines on supercomputing & cloud resources with minimal user overhead or expertise even to novice computing users. In this paper, we describe the Indiana University Trident software project, and the Scalable Compute Archive (SCA) systems built and deployed using Trident. In particular, we describe the evolution of our software design starting with the One Degree Imager Portal, Pipeline, and Archive (ODI-PPA) science gateway [1], offshoot systems we put together using its design, and more recently a brand new microservice suite we designed and developed from scratch. The ODI-PPA science gateway was initially refactored toward two other Astronomy portals (Globular Cluster Systems and Mayal 4m Spectra Data Archive; and another prototype for Ludwig Maximilian University's Wide Field Imager), and an Electron Microscopy portal (IU Electron Microscopy Center Data Archive) these are briefly described in section 2, as are the distinct advantages and drawbacks of the overall system design of the original codebase.

To overcome drawbacks of the original setup, we re-architected our system's overall design from scratch leading to the brand new Indiana University Trident project and its software suite. Trident offers a comprehensive web- and cloud-based microservice software suite that enables the straight forward deployment of highly customized Scalable Compute Archive (SCA) systems – including extensive visualization and analysis capabilities – with minimal amount of additional coding. Trident seamlessly scales up or down in terms of data volumes and computational needs, and allows feature sets within a web user interface to be quickly adapted to meet individual project requirements. Domain experts only have to provide code or business logic about handling/visualizing their domain's data products and about executing their pipelines and application workflows.

Trident uses a microservices architecture, and is made up of light-weight services connected by a REST API and/or a message bus; web interface components built using NodeJS [2], AngularJS [3], and HighCharts [4] JavaScript libraries, and backend services written in NodeJS, and Python. The Trident microservice software suite consists of (1) a Workflow execution framework to integrate, deploy, and execute workflows (2) a Progress service to monitor workflows and sub-workflows (for eg., a complex pipeline data processing job that executes several steps on a set of images) (3) ImageX, an interactive image visualization service [5] (3) an authentication service (4) a Data service that handles archival, staging and serving of data products, and (5) several others including a Notifications service under development. Trident SCA systems leverage supercomputing and storage resources at Indiana University but can be configured to make use of any cloud/grid resource, from local work-stations/servers to (inter)national supercomputing and cloud facilities such as XSEDE. The architecture and the existing prototypes are described in detail in section 3.

2. INITIAL PROJECT EVOLUTION OUT OF ODI-PPA

The One Degree Imager Portal, Pipeline, and Archive (ODI-PPA) science gateway has been in full operations since Oct 2012, and has served as a web science gateway providing astronomers a modern web interface that acts as a single point of access to their data, and rich computational and visualization capabilities. ODI-PPA was designed to be a compute archive that had built-in frameworks including: (1) Collections that allow an astronomer to create logical collations of data products intended for publication, further research, instructional purposes, or to execute data processing tasks (2) Image Explorer and Source Explorer, which together enable real-time interactive visual analysis of massive astronomical data products within an HTML5 capable web browser, and overlaid standard catalog and Source Extractor-generated source markers (3) Workflow framework which enables rapid integration of data processing pipelines on an associated compute cluster and users to request such pipelines to be executed on their data via custom user interfaces. Figure 1, borrowed from the summary paper about ODI-PPA [1], shows the breadth of functionality and features provided by the ODI portal.

We refactored ODI-PPA and deployed several offshot systems described below.

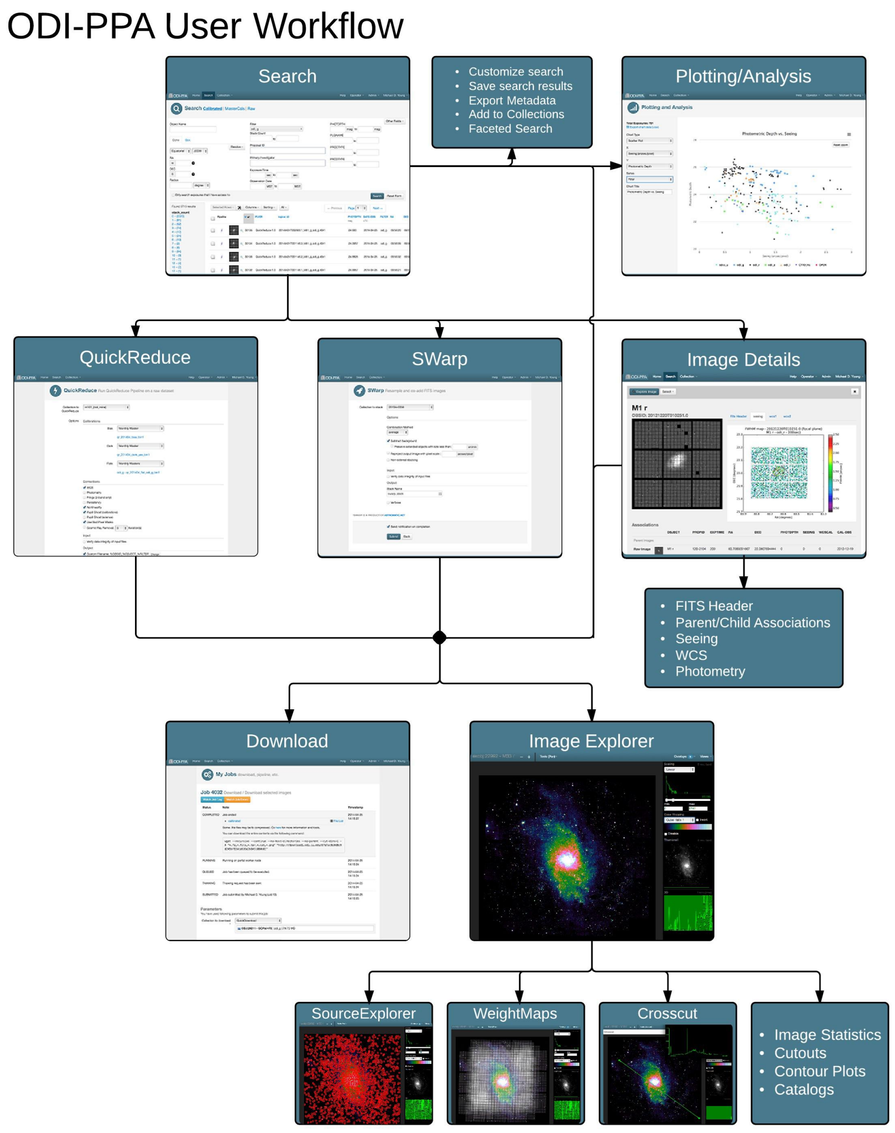


Figure 1. ODI-PPA Functionality Chart

2.1 EMC-SCA

The IU Electron Microscopy Center Scalable Compute Archive (EMC-SCA) is the first offshoot project that stemmed out of ODI-PPA. It has a much more restrictive feature set with the primary objective being to serve as an archive and search/download portal for EMCenter users. For more information, see [6].

2.2 GCS-SCA

The Globular Cluster Systems SCA is a scientific portal and archive for extragalactic globular cluster systems data with a modern and intuitive web interface. It provides public access to the survey results including final stacked mosaic images of the target galaxies. The astrometric and photometric data for thousands of identified globular cluster candidates as well as for all point sources detected in each field, are indexed and made searchable. Where available, spectroscopic follow-up data are paired with the candidates. Advanced imaging tools enable users to overlay these cluster candidates and other sources on the mosaic images within the portal, while metadata charting tools allow users to rapidly and seamlessly plot the survey results for each galaxy and the data for hundreds of thousands of individual sources. The data taken by IU researchers over several years are from a wide-field imaging survey of the globular cluster populations of a sample of giant spiral, S0, and elliptical galaxies with distances of 10-30 Mpc using the mosaic CCD cameras on the WIYN 3.5-m and Kitt Peak 4-m telescopes. For more information, see [7].

2.3 SpArc-SCA

The Spectra Archive SCA portal was developed with the intention of providing long-term storage and accessibility to a collection of observations gathered on the Kitt Peak National Observatorys Mayall 4m telescope taken with the Fourier Transform Spectrometer (FTS) during the period from 1975-1995. Nearly 120 refereed papers that include FTS data have been published since 1978, including nearly 20 refereed papers since 2000. The bulk of the observational data was originally stored on 9-track magnetic tape in a format then in use at the National Solar Observatory. In the late 1990s, the original data on each of 60+ magnetic tapes were converted to ASCII card image format files. While the data were preserved, accessibility was limited and tedious. The data in this archive represents a valuable collection of historical observations, useful for long-term variability studies, samples of precursor objects, or other instances where the value of the data is only apparent in hindsight. Each of the observations, including low resolution images and the forward and backward FTS scans were converted to FITS format. The original metadata are preserved in the FITS headers. For more information, see [8].

2.4 Original Design Philosophy and Choices

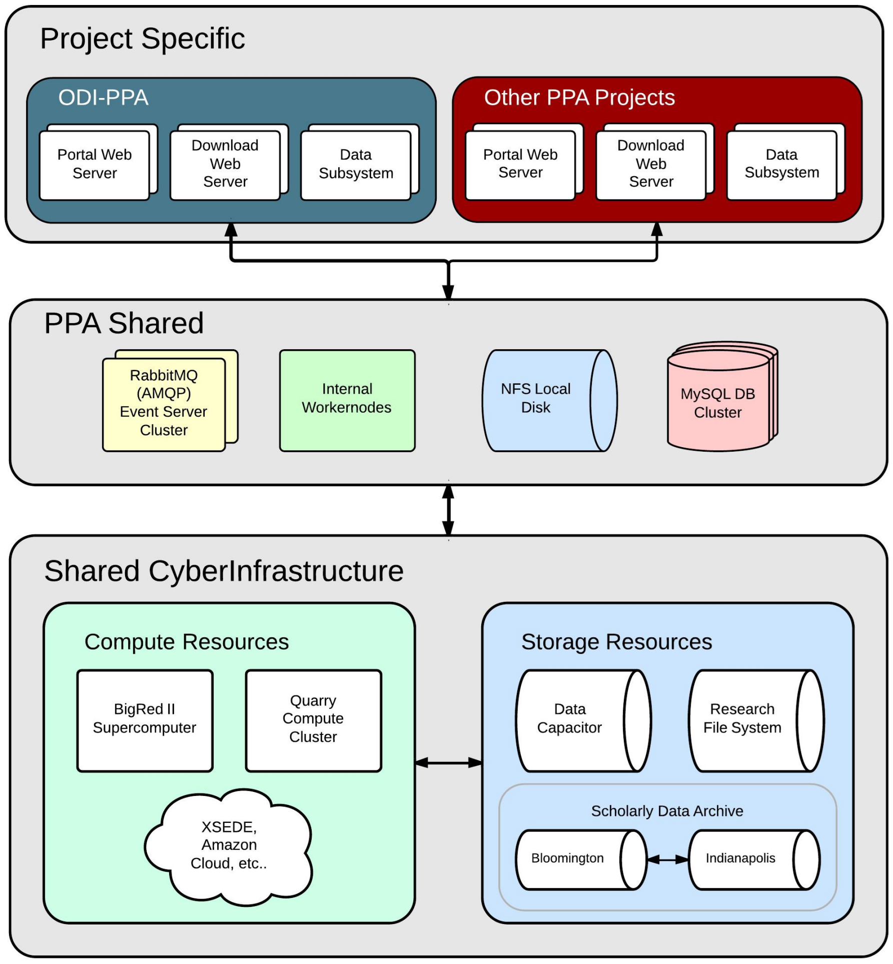


Figure 2. ODI-PPA Design Schematic

Figure 2, also borrowed from the ODI-PPA summary paper, shows our original overall design philosophy. We had deployed distributed systems that communicated with each other via Advanced Message Queue Protocol (AMQP) messages or REST APIs. In particular, we had made a conscious attempt to eschew the tarball/rpm model in which a full-fledged system is developed for one project in a certain domain, and then other groups in the same domain attempt to that deploy that product as a package with minimal ability to modify core design or configuration options (for eg. Metadata schemas/applications, workflow user interfaces). We also placed a significant emphasis on usability and modern user interfaces.

When we began our first offshoot project, EMC-SCA, we cloned the ODI-PPA GIT repository, and immediately began refactoring various parts of the system that needed to precisely designed for the microscopy users needs while also stripping out advanced functions available on ODI-PPA but not relevant to EMCenter users. GCS-SCA continued this approach with even more unique refactored parts and new features including the search interface builder not available on the ODI or EMCenter systems. SpArC-SCA in turn is an offshoot of GCS-SCA.

Scaling Up/Down: Figure 3 shows a visual representation of how our original Trident design was able to seamlessly scale up or down in terms of archive size, compute processing time requirement, or database size. The currently operational ODI-PPA serving the upgraded ODI instrument with 30 OTA detectors (i.e. larger data sizes) is the largest of the projects in terms of archive size, computing needs, and database size whereas SpArc-SCA is barely visible on the plot owing to its small archive size, database size, and computing needs. GCS-SCA has larger database while EMC-SCA is a mid-range and simple Archive/Search/Download system.

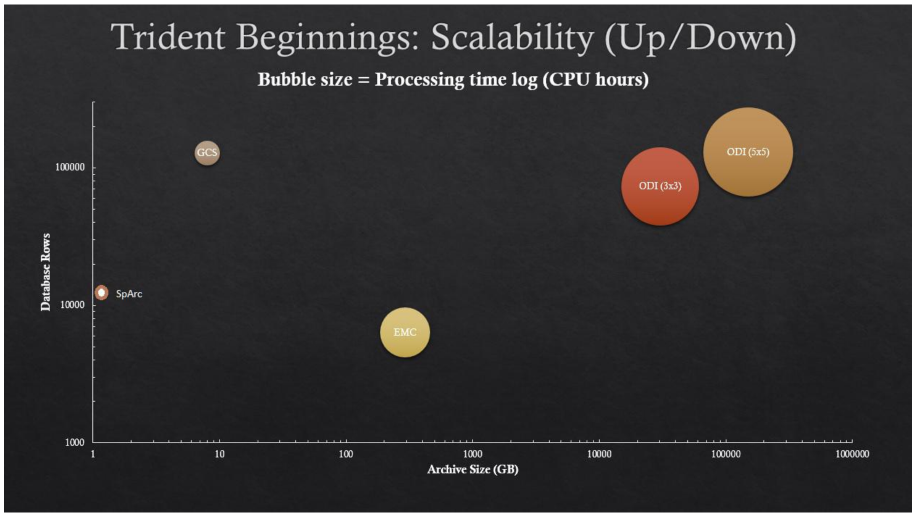


Figure 3. Original Trident Design - Seamless Scaling Up/Down

Technology Stack: The original Trident-SCA systems used a combination of the following software applications/libraries:

- Linux KVM to run Virtual Machines for individual services (for eg. Portal, download servers)
- Apache web server
- PHP/Zend for web application + Javascript libraries including
- Highcharts
- AngularJS
- jQueryUI
- IU TileViewer

- MySQL database
- RabbitMQ Event Server

Drawbacks of Original Design: The design philosophy described above allowed us to deploy precise and custom solutions for each of our stakeholders needs. It also allowed us to share some common services like the MySQL database cluster, the RabbitMQ event server cluster, an NFS file server, and compute/worker nodes. However, our design philosophy had the drawbacks listed below. In the rest of this paper we describe how we started to redesign Trident-SCA systems from scratch.

- It was predicated on Virtual Machine deployments for each distributed service and required a large number of VMs. At one point, and considering production, test, and development service stacks, we had more than 50 VMs running on our hardware resources. Maintaining these VMs was a non-trivial exercise especially when additionally considering a part-time systems administrator.
- The divergence of code between the projects based on separate project level GIT repositories made it difficult to cross-pollinate new features or bug fixes across projects, which was another significant drawback.

3. NEW TRIDENT ARCHITECTURE AND PROTOTYPES

Beginning in fall 2015, we began brainstorming ways to overcome the drawbacks of our prior design. We also recognized that the industry is moving in a different directions with less emphasis on VMs and more emphasis on containers & microservices. We ended up starting over from scratch, and came up with a brand new software design that embraces as many of the modern best practices as possible (more on this in section 3.3). The newly redesigned Trident continues to eschew the tarball/rpm model, and still places significant emphasis on usability and modern user interfaces but it additionally is:

- Predicated on precise microservices that deliver one function or a narrow band of functionality in an extremely reliable manner. These are hosted in their GITHUB repository, and owned by the person who originally wrote the code. GIT pull requests allow for code updates and bug fixes. Thus projects and SCA system deployments share microservices, not just service endpoints like the original design.
- Uses Docker containers, and therefore requires a lot fewer VMs with usually better specifications (CPU, memory, etc.). This allows for easier maintenance and scaling.
- Precisely demarcates shared functionality (for eg. data archival/staging, authentication) vs. domain/project specific functionality (for eg., precise workflows, user interfaces for those workflows, handling and visualization of data products.)
- Is workflow-centric, and requires precise code or business logic to be provided by a domain/project expert.

3.1 Separating Shared Components from Domain/Project Specific Components

Trident attempts to precisely demarcate between components that can be shared by multiple projects/deployed systems and those that are unique and custom for each domain/project. A more visual representation of the description in this section is available in section 4. Below we describe several sharable components including:

- **Workflow Execution Framework:** Handles mapping of resources to pipelines, data transfer between resources, tracking of workflow job status, etc. This component is capable of embedding more advanced workflow middleware if necessary, and also provides common user interface elements for the aforementioned functions.
- **Progress Service:** Allows tracking of progress of arbitrary tasks and subtasks.
- **Data Service**: Handles common data movement tasks like archiving to tape, staging to disk, serving files off a web server.

- **AuthN/AuthZ:** Enables authentication and authorization including group and role mappings to each user, and associated privilege to execute certain functions within a deployed web portal.
- **Generic Input Data Handling:** Allows basic input data handling including file uploads (via drag and drop), staging from a public resource (via a URL), staging off tape or another disk resource, etc.

Trident requires the following to be provided by a domain expert in the form of application/code modules on GITHUB or at least the business logic:

- **Application/Pipeline Execution Instructions:** Information about what applications/pipelines are necessary, and instructions on executing them.
- **Workflow-UI:** Defines the user interface elements both from the input side and the output data products side that would be relevant to the community/project users.
- **Data Product:** Defines how input data as well as output data are to be handled including validation steps.
- **Data Product-UI:** Defines how input data as well as output data are to be displayed or visualized.

A more detailed description of the workflow execution framework, etc. is beyond the scope of this paper, and will be published in an upcoming paper.

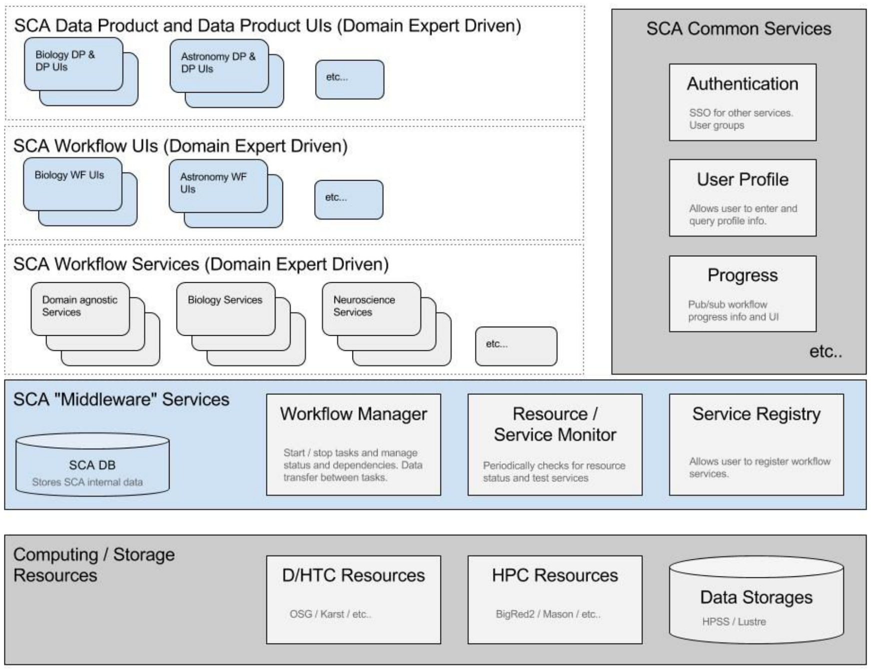

SCA Data Product and Data Product UIs (Domain Expert Driven)
Biology DP & DP UIs
Astronomy DP & DP UIs
etc...
SCA Workflow UIs (Domain Expert Driven)
Biology WF UIs
Astronomy WF UIs
etc...
SCA Workflow Services (Domain Expert Driven)
Domain agnostic Services
Biology Services
Neuroscience Services
etc...
SCA Common Services
Authentication
SSO for other services. User groups
User Profile
Allows user to enter and query profile info.
Progress
Pub/sub workflow progress info and UI
etc..
SCA "Middleware" Services
SCA DB
Stores SCA internal data
Workflow Manager
Start / stop tasks and manage status and dependencies. Data transfer between tasks.
Resource / Service Monitor
Periodically checks for resource status and test services
Service Registry
Allows user to register workflow services.
Computing / Storage Resources
D/HTC Resources
OSG / Karst / etc..
HPC Resources
BigRed2 / Mason / etc..
Data Storages
HPSS / Lustre

Figure 4. New Trident Design Schematic

3.2 Fully Functional Command Line API

The redesigned Trident software suite does not require a potential third party user to depend on any of the user interface elements we provide if they so prefer. It has a comprehensive Command Line API (CLI) that anyone can use to interface with their own UI components. For example:

```
# sca login john@iu.edu
# sca task status 123456
```

A more detailed description of the CLI is beyond the scope of this paper, and will be published in an upcoming paper.

3.3 Technology Stack & Best Practices

In this section, we briefly outline the technologies and best practices embraced by our new software design.

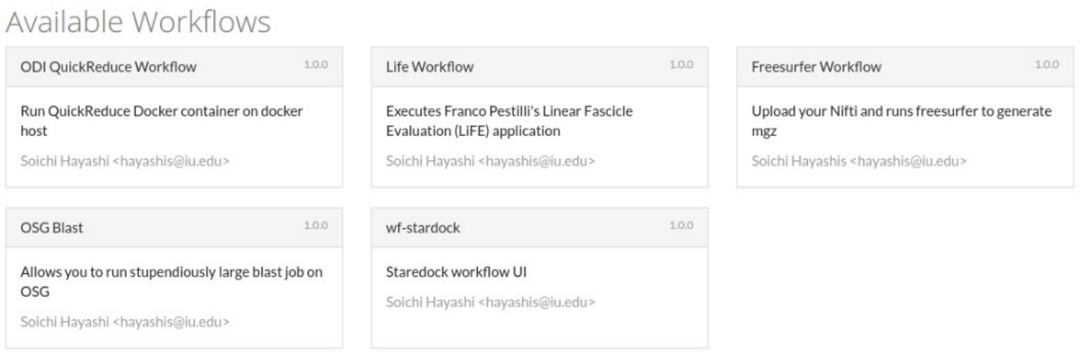
- **Docker** has become one of the most widely used container technologies, and has 100,000+ applications ready to be deployed on most modern Operating Systems. We use Docker to run both commodity services and home-grown containers including DockQR (a Dockerized version of the QuickReduce pipeline) [9].
- **Javascript** has evolved from an unstable web front end utility language to become one of the most widely used programming languages (for eg. it has become the most used programming language on GITHUB [10]) with a wide variety of platforms, libraries, and utilities. Among other libraries, in particular we use the following heavily:
  - **NodeJS** to write both frontend and backend applications.
  - **AngularJS** for frontend web interface functionality
  - **HighCharts** for dynamic plotting of metadata, etc.
  - **OpenSeadragon** used within ImageX to render tile pyramids
- **Logistics**
  - **GITHUB** to host all code, application modules, etc.
  - **Jenkins & TravisCI** for automated build testing
- **Deployment & Monitoring**
  - **Ansible** for reproducable deployment of services and hosts
  - **Linux Virtual Service (LVS)** for High Availability and service forwarding
  - **Nginx** for all basic web server needs
  - **PM2** for starting/stopping/restarting microservices and application scripts
  - **ELK Stack + Sensu + Uchiwa** for log processing, host & service monitoring
  - **Slack** for alerts when certain events including downtimes occur
- **MongoDB** for non-RDBMS database needs

A more detailed description of the technologies and best practices used by Trident is beyond the scope of this paper, and will be published in an upcoming paper.

4. TRIDENT PROTOTYPES

In this section, we will briefly describe the prototype workflows we have developed at this time. We use limited verbal description and rely on screenshots to convey the exact nature of functionality more clearly. We start with generic user interfaces common to most workflows including:

- **Authentication:** Trident provides a versatile authentication/authorization component that allows typical login mechanisms like username/password, Google, Facebook, etc. as well as robust authorization mappings to roles, levels, groups, etc. A screenshot is shown in figure 6.



Available Workflows

ODI QuickReduce Workflow 1.0.0
Run QuickReduce Docker container on docker host
Soichi Hayashi <hayashis@iu.edu>

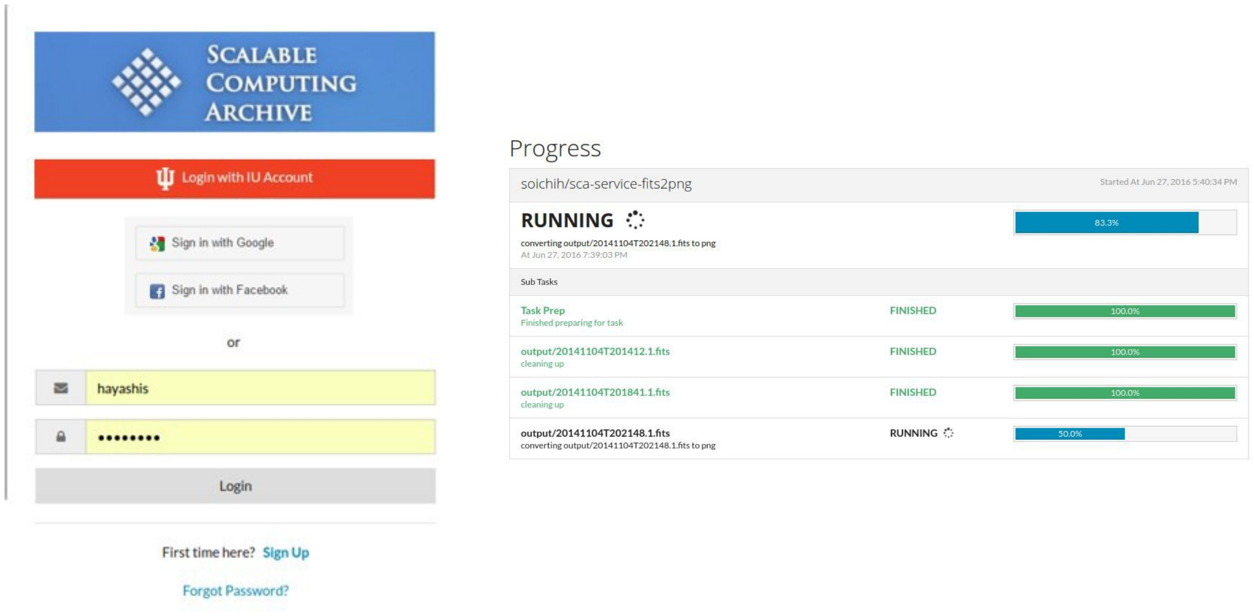
Life Workflow 1.0.0
Executes Franco Pestilli's Linear Fascicle Evaluation (LiFE) application
Soichi Hayashi <hayashis@iu.edu>

Freesurfer Workflow 1.0.0
Upload your Nifti and runs freesurfer to generate mgz
Soichi Hayashis <hayashis@iu.edu>

OSG Blast 1.0.0
Allows you to run stupendiously large blast job on OSG
Soichi Hayashi <hayashis@iu.edu>

wf-stardock 1.0.0
Staredock workflow UI
Soichi Hayashi <hayashis@iu.edu>

Figure 5. Prototype - List of available workflows



hayashis

••••••••

Progress

soichih/sca-service-fits2png — Started At Jun 27, 2016 5:40:34 PM

RUNNING 83.3%
converting output/20141104T202148.1.fits to png
At Jun 27, 2016 7:39:03 PM

Sub Tasks

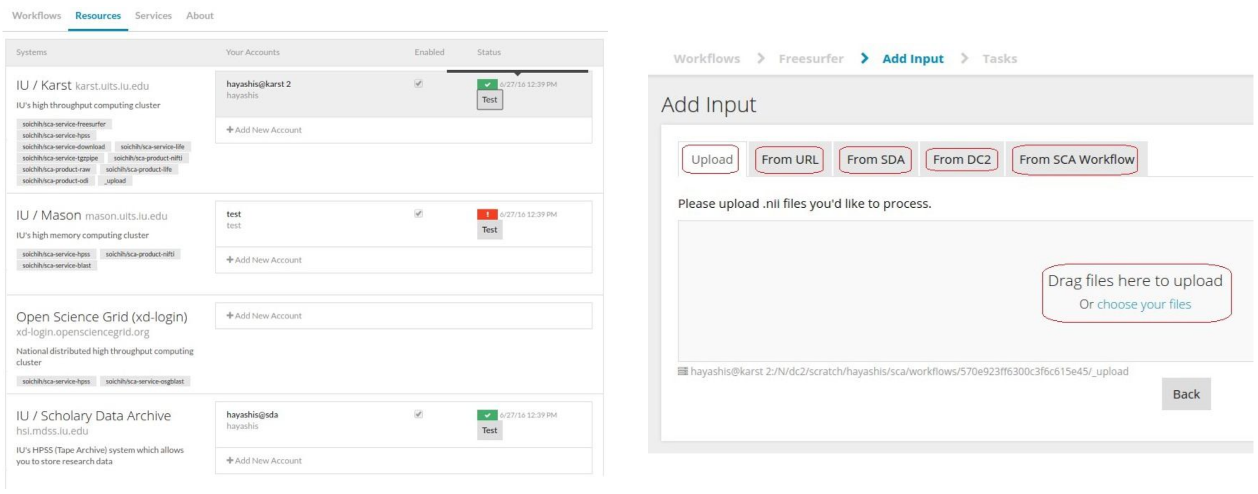
Task Prep Finished preparing for task	FINISHED	100.0%
output/20141104T201412.1.fits cleaning up	FINISHED	100.0%
output/20141104T201841.1.fits cleaning up	FINISHED	100.0%
output/20141104T202148.1.fits converting output/20141104T202148.1.fits to png	RUNNING	50.0%

Figure 6. Prototype - Login UI (left) and Workflow Progress (right)

- **Resource Setup for User and Status:** The Trident SCA framework has a common resource setup/status monitoring component including a user interface. This allows any registered user to map one or more known resource (or even unknown resources, for eg., a cluster in their institution) for use by the SCA workflows. SSH key pairs or Kerberos keytabs are the preferred method of seamkess authentication setup but the Resource module can plug in other methods as a user may see fit. This component also does basic status check-ups of the resources for each user, so if a certain resource is down for maintenance then it is automatically removed from workflow submissions till it comes back up. A screenshot is shown in figure 7.

- **Generic File Uploads / Staging:** While some scientific workflows may require custom data selection (for eg., metadata search based input selection used by the ODI QuickReduce workflow described below), many workflows typically have common input data staging needs. Trident has a reusable component including the user interface that allows the user to upload data from their computer (via drag and drop or by selecting one or more files), stage data from a public resources (via a URL), stage data for registered archival or disk systems. A screenshot is shown in figure 7.

- **Generic Workflow Progress:** Trident provides a simple and yet powerful Progress service including its



Systems	Your Accounts	Enabled	Status
IU / Karst karst.uits.iu.edu IU's high throughput computing cluster soichih/sca-service-freesurfer soichih/sca-service-hpss soichih/sca-service-download soichih/sca-service-life soichih/sca-service-tgzpipe soichih/sca-product-nifti soichih/sca-product-raw soichih/sca-product-life soichih/sca-product-odi _upload	hayashis@karst 2 hayashis + Add New Account	✓	✓ 6/27/16 12:39 PM Test
IU / Mason mason.uits.iu.edu IU's high memory computing cluster soichih/sca-service-hpss soichih/sca-product-nifti soichih/sca-service-blast	test test + Add New Account	✓	! 6/27/16 12:39 PM Test
Open Science Grid (xd-login) xd-login.opensciencegrid.org National distributed high throughput computing cluster soichih/sca-service-hpss soichih/sca-service-osgblast	+ Add New Account		
IU / Scholary Data Archive hsi.mdss.iu.edu IU's HPSS (Tape Archive) system which allows you to store research data	hayashis@sda hayashis + Add New Account	✓	✓ 6/27/16 12:39 PM Test

Add Input

Please upload .nii files you'd like to process.

Drag files here to upload
Or choose your files

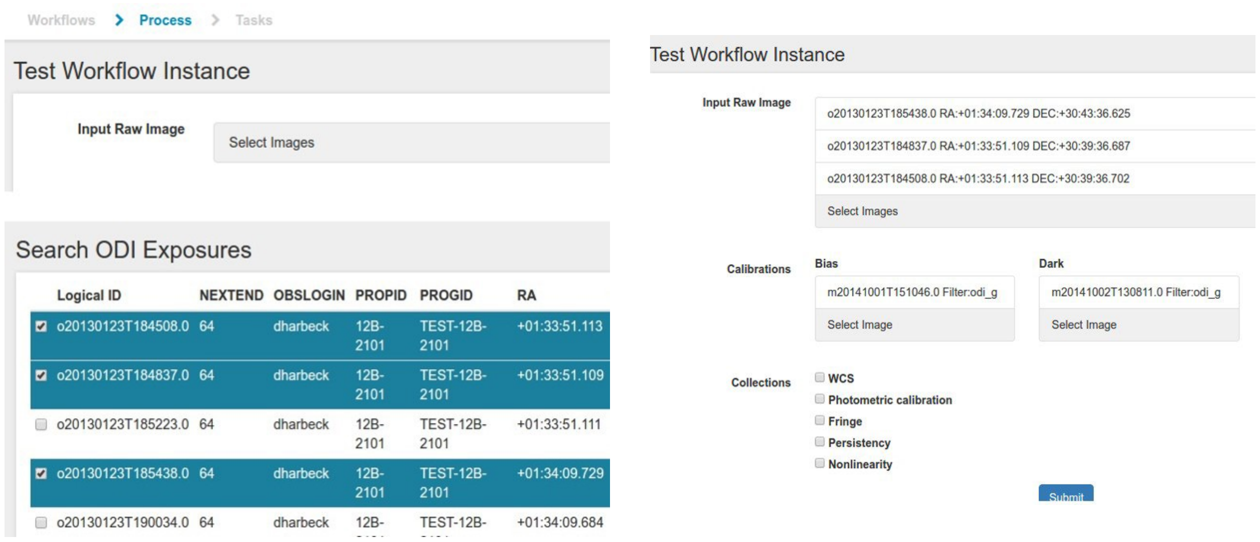
hayashis@karst 2:/N/dc2/scratch/hayashis/sca/workflows/570e923ff6300c3f6c615e45/_upload

Figure 7. Prototype - Generic Input Data Handling (left0 and Resource Setup/Status (right)

own user interface that can be used to track arbitrary levels of progress of workflow tasks and subtasks. A screenshot is shown in figure 6.

4.1 Astronomy: QuickReduce

The QuickReduce pipeline has been an integral part of the ODI-PPA system providing operator and user calibrated data as well as enabling SWarp stacking of images. We have an end-to-end prototype which allows data selection via raw metadata search, calibration frame selection based on lunation, month, observing proposal, etc., execution of the pipeline with preferred parameters, and finally allows the user to visualize output data products using ImageX. Example screenshots of these steps are shown in figures 8 and 9



Test Workflow Instance

Input Raw Image: Select Images

Search ODI Exposures

	Logical ID	NEXTEND	OBSLOGIN	PROPID	PROGID	RA
✓	o20130123T184508.0	64	dharbeck	12B-2101	TEST-12B-2101	+01:33:51.113
✓	o20130123T184837.0	64	dharbeck	12B-2101	TEST-12B-2101	+01:33:51.109
	o20130123T185223.0	64	dharbeck	12B-2101	TEST-12B-2101	+01:33:51.111
✓	o20130123T185438.0	64	dharbeck	12B-2101	TEST-12B-2101	+01:34:09.729
	o20130123T190034.0	64	dharbeck	12B-	TEST-12B-	+01:34:09.684

Test Workflow Instance

Input Raw Image
o20130123T185438.0 RA:+01:34:09.729 DEC:+30:43:36.625
o20130123T184837.0 RA:+01:33:51.109 DEC:+30:39:36.687
o20130123T184508.0 RA:+01:33:51.113 DEC:+30:39:36.702
Select Images

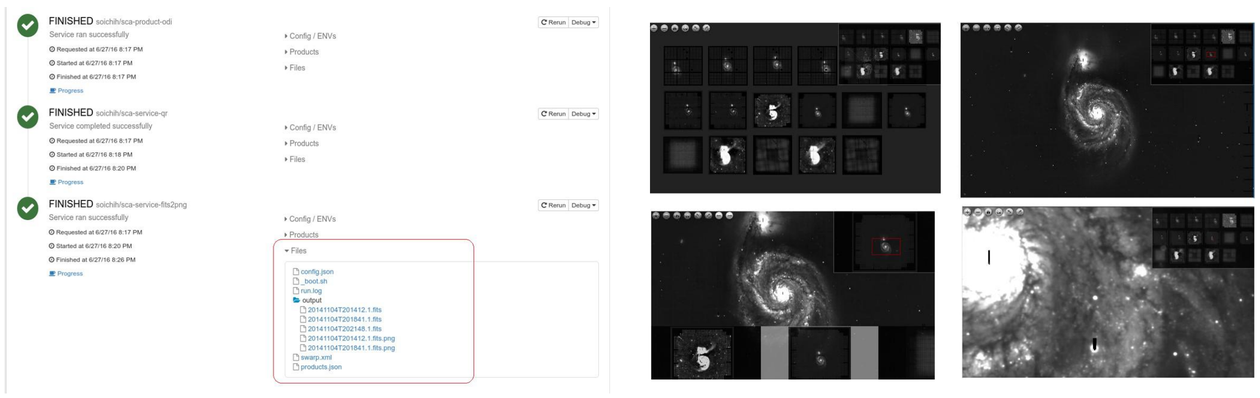
Calibrations
Bias: m20141001T151046.0 Filter:odi_g
Select Image
Dark: m20141002T130811.0 Filter:odi_g
Select Image

Collections
☐ WCS
☐ Photometric calibration
☐ Fringe
☐ Persistency
☐ Nonlinearity

Figure 8. Prototype - Input Data Selection by Metadata Search (left) and Submit Options (right)

4.2 Astronomy: StarDock

StarDock is a new research project at IU that aims to address the problem of surging data volumes and the increasing unfeasibility of transferring astronomical datasets to the local systems of individual scientists for


FINISHED soichih/sca-product-odi
Service ran successfully
Requested at 6/27/16 8:17 PM
Started at 6/27/16 8:17 PM
Finished at 6/27/16 8:17 PM

FINISHED soichih/sca-service-qr
Service completed successfully
Requested at 6/27/16 8:17 PM
Started at 6/27/16 8:18 PM
Finished at 6/27/16 8:20 PM

FINISHED soichih/sca-service-fits2png
Service ran successfully
Requested at 6/27/16 8:17 PM
Started at 6/27/16 8:20 PM
Finished at 6/27/16 8:26 PM

```
config.json
_boot.sh
run.log
output
    20141104T201412.1.fits
    20141104T201841.1.fits
    20141104T202148.1.fits
    20141104T201412.1.fits.png
    20141104T201841.1.fits.png
swarp.xml
products.json
```

Figure 9. Prototype - Completed QuickReduce Workflow Output incl. Simple File Listing (left) and ImageX View (right)

analysis. It leverages the Docker container application virtualization software, along with a suite of commonly used astronomy applications, and allows users to configure a container with their own custom software and analysis tools. StarDock system moves the users container to the data, and exposes the requested dataset, allowing users to safely and securely process their data without needlessly transferring hundreds of gigabytes. A sample screenshot of a websh based SSH terminal acquired via StarDock is shown on figure 10.

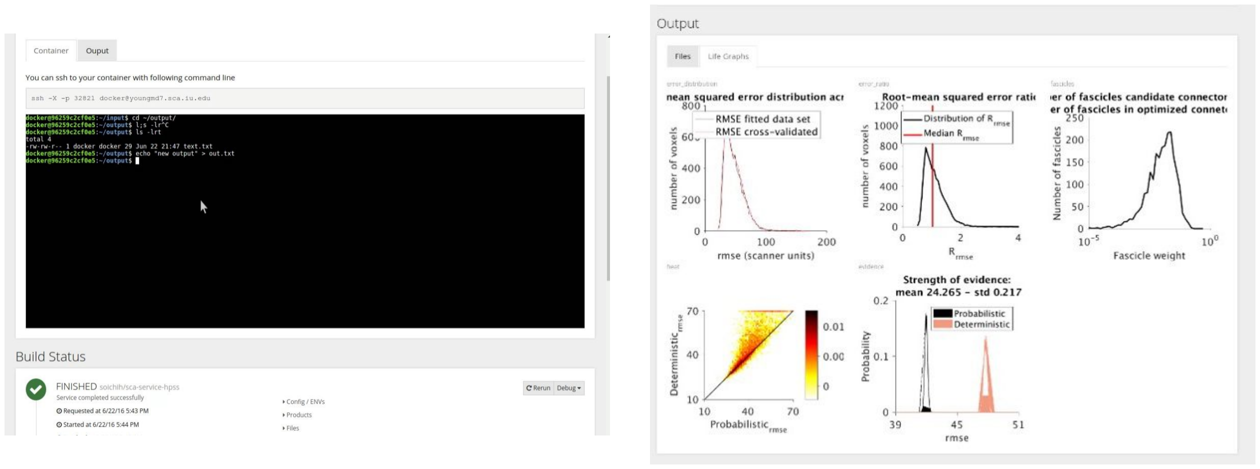
4.3 Other Domains

Apart from the astronomy prototypes described above, we have also deployed three other applications from three different domains as prototype workflows:

- FreeSurfer: An application widely used by the radiology community to process DICOM and NIFTI formatted images.
- BLAST: An application commonly used by the Bioinformatics community to do sequence comparison.
- LIFE: A neurology application developed by an IU faculty member. An example screenshot of the output of this workflow is shown in 10.
- Desktop Snapshot Tool: A snapshot tool being developed to snapshot directories into a tape archive from within the IU Karst Desktop system is being geared to use the Trident Command Line API on the backend.
- GIS: The IU GIS website uses Trident backend components to stage data off a tape archive, and serve them to users.

5. CONCLUSION & FUTURE WORK

The Trident project aims to contribute toward enabling efficient knowledge discovery and scientific advancement in the Big Data era with new and equally efficient tools: modern user interfaces for searching, identifying and viewing data online without direct access to the data; tracking of data provenance; searching, plotting and analyzing metadata; interactive visual analysis, especially of (time-dependent) image data; and the ability to execute pipelines on supercomputing & cloud resources with minimal user overhead or expertise even to novice computing users. It offers a comprehensive web- and cloud-based microservice software suite that enables the straight forward deployment of highly customized Scalable Compute Archive (SCA) systems – including extensive visualization and analysis capabilities – with minimal amount of additional coding. Trident manages to seamlessly scale up or down in terms of data volumes and computational needs, and allows feature sets within a web user interface to be quickly adapted to meet individual project requirements. It has a distinct advantage that ddomain experts only have to provide code or business lodgic about handling/visualizing their domain's data products and about executing their pipelines and application workflows. The Trident microservice software



Container | Ouput

You can ssh to your container with following command line

```
ssh -X -p 32821 docker@youngmd7.sca.iu.edu
```

```
docker@96259c2cf0e5:~/input$ cd ~/output/
docker@96259c2cf0e5:~/output$ l;s -lr^C
docker@96259c2cf0e5:~/output$ ls -lrt
total 4
-rw-rw-r-- 1 docker docker 29 Jun 22 21:47 text.txt
docker@96259c2cf0e5:~/output$ echo "new output" > out.txt
docker@96259c2cf0e5:~/output$
```

Build Status

FINISHED soichih/sca-service-hpss
Service completed successfully
Requested at 6/22/16 5:43 PM
Started at 6/22/16 5:44 PM

Figure 10. Prototype - Results of StarDock (left) and LIFE (right) workflows

suite consists of (1) a Workflow execution framework (2) a Progress service (3) the ImageX interactive image visualization service (4) an authentication service (5) a Data service that handles archival, staging and serving of data products, and (6) several others under development.

Future work is expected to be geared toward making existing microservices more robust and highly available while transitioning various domain/project prototypes to production services.

REFERENCES

[1] Gopu, A., Hayashi, S., Young, M. D., Harbeck, D. R., Boroson, T., Liu, W., Kotulla, R., Shaw, R., Henschel, R., Rajagopal, J., Stobie, E., Knezek, P., Martin, R. P., and Archbold, K., "Odi - portal, pipeline, and archive (odi-ppa): a web-based astronomical compute archive, visualization, and analysis service," (2014).

[2] "Nodejs." https://nodejs.org.

[3] "Angularjs." http://angularjs.org.

[4] HighsoftAS, "Highcharts." http://www.highcharts.com.

[5] Hayashi, S., Gopu, A., Kotulla, R., and Young, M. D., "Imagex: New and improved image explorer for astronomical images and beyond.," (2016).

[6] Morgan, D., Gopu, A., Young, M. D., and Hayashi, S., "Emc-sca: Organizing, managing and searching large collections of images: A new resource to handle high-throughput imaging," (2014).

[7] Young, M. D., Rhode, K., and Gopu, A., "A science portal and archive for extragalactic globular cluster systems data," (2015).

[8] Pilachowski, C., Hinkle, K., Brokaw, H., Young, M. D., and Gopu, A., "Preservation of 20 years of spectrographic data," (2016).

[9] Kotulla, R., Gopu, A., and Hayashi, S., "Rabbitqr: Fast and flexible big data processing at lsst data rates using existing, shared-use hardware.," (2016).

[10] "Language trends on github." https://github.com/blog/2047-language-trends-on-github.