Federated Observational and Simulation Data in the NASA Center for Climate Simulation Data Management System Project

John L. Schnase[1], Glenn Tamkin[2], David Fladung[2], Scott Sinno[2], and Roger Gill[2]

[1] Office of Computational and Information Science and Technology
[2] NASA Center for Climate Simulation (NCCS)
NASA Goddard Space Flight Center
Greenbelt, MD 20771

Abstract

The NASA Center for Climate Simulation (NCCS) plays a lead role in meeting the computational and data management requirements of climate modeling and data assimilation. Scientific data services are becoming an important part of the NCCS mission. The NCCS Data Management System (DMS) is a key element of NCCS's technological response to expanding data services. In DMS, we are using the Integrated Rule-Oriented System (iRODS) to integrate disparate data collections into a federated platform upon which a wide range of data services can be implemented. Work to date has demonstrated the effectiveness of iRODS in managing a large-scale collection of observational data, in managing model output data in a cloud computing context, and in managing NCCS-hosted data products that are published through community-defined services such as the Earth System Grid (ESG).

***Index Keyword Terms*—**iRODS, data services, archive management

1. Introduction

The NASA Center for Climate Simulation (NCCS) provides large-scale compute engines, analytics, data sharing, long-term storage, networking, and other high-end computing services designed to meet the specialized needs of the Earth science communities. By doing so, NCCS brings NASA observational and model data products to climate research carried out by a wide range of national and international organizations [1, 2].

Over the past year, we have examined the potential of iRODS, the Integrated Rule-Oriented Data System, as a means of integrating and delivering scientific data services to the communities we serve. We call this effort the Data Management System project, and it has resulted in the NCCS Data Management System (DMS) — a testbed collection of independent iRODS data systems comprising observational and simulation data. We have used this opportunity to learn about iRODS and understand how the technology might further our mission. In particular, we have tried to understand if iRODS can provide a comprehensive, federated platform upon which to build a collection of scientific data services tailored to the needs of our customers.

In the following sections, we describe our experiences with the DMS project, including motivating factors behind the effort, rationale for focusing on iRODS, implementation details, lessons learned, and our future plans regarding scientific data services in the NCCS.

2. Background

A key challenge for the Earth science community is to find and access massive amounts of observational and model data for use in climate and weather studies. As Earth science applications grow in complexity and resolution, they are requiring unprecedented access to large amounts of distributed data — on the order of terabytes and petabytes. In most cases, the data are geographically distributed, difficult to find, and even more difficult to access and understand [3, 4].

NCCS's mission is expanding to include a broader range of data and information services. First, NCCS's two major customers, NASA's Global Modeling and Assimilation Office (GMAO) and the Goddard Institute for Space Studies (GISS) will be contributing products to the Intergovernmental Panel on Climate Change (IPCC) Fifth Assessment Report (AR5) [5, 6]. IPCC is coordinating a team of 831 climate change experts working throughout the world to produce AR5, which will be published between 2013 and 2014 [7]. These activities require that the NCCS provide the data management services and analytical tools necessary for GMAO and

GISS to conduct their work and support the data publication requirements of the IPCC.

Another requirement results from the tie that exists between NASA modeling efforts and satellite missions: observational data provide the means for evaluating and improving climate models. There is growing interest in bringing the climate modeling and observational communities together to work toward the goal of integrating model outputs and observational data [8]. Goddard Space Flight Center, being home to GMAO and many of NASA's Earth observing missions, is uniquely positioned to contribute to this effort, and these observational/simulation data integration activities are becoming an important part of NCCS's data services mission.

Finally, we recognize that computing requirements for Earth system modeling will increase significantly in the coming years [9]. We also recognize that high-end computing requires more than increased speed. Rapid access to large volumes of heterogeneous and geographically distributed data will be needed along with enhanced archiving capabilities, enhanced analysis capabilities, and the capacity to manage all aspects of high-performance scientific workflows. NCCS must keep pace with innovations that can address these needs.

It is against this backdrop that the NCCS began looking at iRODS as a potential element in our technological and organizational response to changing demands.

3. NCCS Data Management System

The DMS project has been an effort to learn about iRODS and understand first hand if this technology might provide a comprehensive, end-to-end approach to managing data and data services in the NCCS.

iRODS appealed to us for several reasons. It targets large repositories, large data objects, digital preservation, and integrated complex processing, making it one of the more promising technologies for grid-centric data services for scientific applications [10, 11, 12]. We also liked the fact that its development culture has historic roots in digital libraries, persistent archives, and real-time data systems research, having received support from the National Science Foundation and National Archives and Records Administration.

Our strategy for gaining experience with iRODS was to build four independent iRODS data systems comprising a range of data types and circumstances relevant to our data center. Two of the systems, merra_Zone and yotc_Zone, manage simulation data products; the other two, modis_Zone and isds_Zone, manage observational data products.

Using these data systems as a testbed, we then looked at broader issues of data integration, federation, and generalized data services support. We also tried to understand how iRODS might affect our operations from an organizational perspective by considering the level and type of technical staffing required to support iRODS; the work involved in building, deploying, and maintaining iRODS systems; the work required to implement, test, and maintain domain- and NCCS-specific iRODS micro-services; how we might accommodate new and legacy data; customer impacts, including early adopter and end user support issues; and the financial, political, and cultural issues attached to a major new technology initiative such as this.

In this section, we provide details about how the four iRODS systems were implemented. We assume general familiarity with iRODS; readers wanting to know more will find helpful descriptions on the iRODS website [10].

3.1. modis_Zone: MODIS Earth Observational Data

The goal for modis_Zone was to use iRODS to provide access to Moderate Resolution Imaging Spectroradiometer (MODIS) atmosphere data products. This exercise allowed us to gain experience with iRODS in the setting of a production, flight mission observational data system.

MODIS is a key instrument aboard the NASA Terra and Aqua satellites [13]. Terra's orbit around the Earth is timed so that it passes from north to south across the equator in the morning, while Aqua passes south to north over the equator in the afternoon. Terra MODIS and Aqua MODIS are viewing the entire Earth's surface every one to two days, acquiring data in 36 spectral bands, or groups of wavelengths. These data will improve our understanding of global dynamics and processes occurring on the land, in the oceans, and in the lower atmosphere. MODIS is playing a vital role in the development of validated, global, interactive Earth system models able to predict global change accurately enough to assist policy makers in making sound decisions concerning the protection of our environment.

MODIS Adaptive Processing System (MODAPS) Web is the web interface to the MODIS Data Processing System, or MODAPS [14]. MODAPS generates Level 2 through Level 4 MODIS science products for distribution to the Goddard Earth Sciences Data and Information Services Center (GES DISC) for archival storage and to the MODIS science team for quality control [15]. MODAPS Web is one method used by the MODIS science team to access MODIS science products. The science team may also view reports on the status of data processing with MODAPS Web.

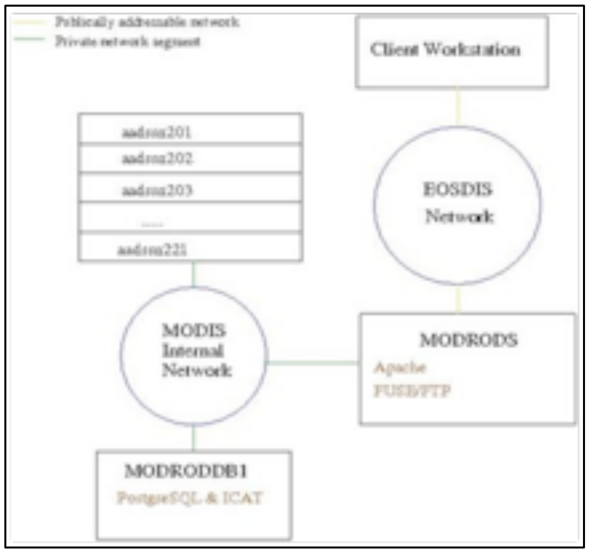
The modis_Zone system, which was deployed on the operational MODAPS servers, consists primarily of the iRODS application, iRODS/iCAT database, and 22 storage nodes. The entire collection of MODIS products was registered. The system was initially opened to MODIS users via the Filesystem in Userspace (FUSE) interface. With FUSE, the iRODS/iCAT database itself serves as the filesystem. As a result, any database updates are reflected in the iRODS collection nearly instantaneously, thus eliminating the need for expensive consistency check jobs between filesystem and database.

3.1.1. Architecture

The modis_Zone implementation is centered around two Dell Poweredge servers, each equipped with the following hardware resources:

- 4x Quad Core X5550 (Nahalem) processors
- 48GB DDR3 memory
- 2x 15k RPM SAS hard drives in a RAID1 (mirrored) array for standard OS
- 4x 15k RPM SAS hard drives in a RAID10 (stripe + mirror) array

As shown in the figure below, one of the servers, modrods, is attached to the EOSDIS network as modrods.modaps.eosdis.nasa.gov in order to function as a public interface for the modis_Zone, AADS (Atmospheric Archived Data Sets). The other, modroddb1, functions as the iCAT server for our AADS zone. Additionally, an iRODS server was installed on all 21 of MODIS's pre-existing Atmospheric Data Storage Nodes, or aadssn nodes. An iRODS daemon running on each of these nodes ties them into our AADS grid.



To mitigate security concerns, all software used for the modis_Zone has been setup to run as a dedicated system user named iguy. This measure establishes a hard degree of privilege separation such that iRODS lacks any write privileges beyond what it absolutely requires. This ensures the safety of the underlying MODIS data in the event of a compromise or serious bug with the iRODS software.

The following table identifies all of the software components used within the standard modis_Zone deployment of iRODS:

Name	Version	Notes
iRODS	2.3	Core iRODS installation. Includes i-commands.
Extrods	1.1.0.1-beta	Officially provided iRODS web UI.
PHP	5.2.14	Required for iRODS web UI.
Apache web server	2.2.15	Required to serve iRODS web UI.
FUSE library	2.7.4	Base FUSE library required for iRODS FUSE interface.
Postgresql	8.4.4	Required RDBMS for iCAT.
UnixODBC	2.2.12	Required for iRODS communication to iCAT.
NCFTP	3.2.3	FTP Service backended by FUSE
CENTOS	5.5	Base Operating System

The standard iRODS configuration was installed on each server with the exception of the extensions described below. A simple custom database, MODCAT, was constructed to stand alongside the iCAT database in order to assist with the task of tracking, scheduling, and controlling registration tasks. MODCAT tables include:

- **active_jobs** - Tracks what jobs are actively running on what systems. Host, pid, start time, number of files registered so far, and total number to be registered are all viewable here.
- **files_in_irods** - Tracks MODIS FileID's that have been successfully registered into iRODS, thereby avoiding unnecessary re-registrations.
- **cfg** - Stores simple configuration values, such as the maximum number of ingest jobs that may execute concurrently.
- **job_log** - A history of all ingest jobs that have been run on the system, retained primarily for purposes of tracking execution times, errors, etc.

A standalone bash script, iController, was written to schedule and control registration of MODIS's large atmospheric data collections. iController accepts the following commands:

- **start** - Starts up iController.sh as a backgrounded daemon. It will invoke up to maxrunning irods_disk_ingest jobs, as defined in the cfg table within the MODCAT database.
- **stop** - Stops iController.sh on the head-node (modrods). This does not effect the remote ingest jobs – they will continue to run to completion. This action is appropriate if the head-node and only the head-node needs to be reset or shutdown.
- **halt_ingest** - Gracefully terminates all active ingest jobs on all remote nodes.
- **pause_ingest** - Pauses all active ingest jobs on all remote nodes. This is useful for quick configuration changes on the iCAT/database side of things that require a postgres restart, or for restarting irodsServer on the iCAT host.

- **resume_ingest** - Un-pause ingests previously paused by the above directive.
- **status** - Displays the status of all active jobs by dumping the contents of the active_jobs table in the MODCAT DB.

When iController starts, it queries the MODCAT active_jobs table to determine if new registration jobs should be executed. If more jobs are to be launched, iController will query the MODAPS distribution database to get a list of eligible aadssn nodes that are marked online; this prevents iController from trying to run registration jobs on a server or filesystem that has been flagged as offline for one reason or another (maint, outage, etc.). iController then randomizes the list of eligible servers and filesystems and begins launching registration jobs on the remote nodes until maxrunning is reached. Upon completion of that goal, it will go dormant for a short period, then repeat the process.

iController invokes the registration jobs by calling another script we authored, irods_disk_ingest. This script performs the task of actually registering (ireg) and metadata-tagging (imeta) files. This script must be passed a filesystem to scan for registering data, and must be executed on the aadssn node locally storing that data. iController handles this aspect of the task, although it could also be done manually. In either case, this ingest script executes as follows:

- Check to ensure no other ingest is already running against the same filesystem (in case a manual run was invoked while iController was already on the case), and if not add a record for itself in the active_jobs table in MODCAT.
- Query the MODAPS distribution database (MODLADS) to get a list of FileIds stored on the target filesystem.
- Compare the list above with what has already been ingested on this filesystem, as recorded in the files_in_irods table of MODCAT, and ignore FileId's we already know to be registered.
- For each FileId, determine the physical disk path, DataDay, CollectionId, and ESDT (product type) of the associated file by querying MODLADS. Use this information to register the file into an appropriate path based on this metadata.
- Query MODLADS for other metadata of interest and use imeta to tag the registered file within the iRODS grid.
- Upon completion of registering all identified target files, remove the related entry for the job from active_jobs table in MODCAT, and create a new one in job_log populated with relevant information, including error count, runtime, successful ingest count, etc.

The process of registering the entire collection of publicly available MODIS atmospheric data products was initially very slow. The following database indexes were added to speed up the registration and search functions:

- customidx_data_main1 on r_data_main using hash (data_path); Improves data registration time and search-time when searching against collections (i.e. Directories).
- customidx_data_main2 on r_data_main (data_name); Improves search-time when searching against data-objects (i.e. Filenames).
- customidx_meta_main1 on r_meta_main (meta_attr_name,meta_attr_value,meta_attr_unit); Vastly improves metadata-tagging time.

The following indexes were also implemented after discussion and in coordination with the iRODS development team. All five vastly improve search-time by metadata attributes/values:

- idx_meta_main2 on r_meta_main (meta_attr_name).
- idx_meta_main3 on r_meta_main (meta_attr_value).
- idx_meta_main4 on r_meta_main (meta_attr_unit).
- idx_objt_metamap5 on r_objt_metamap (meta_id).
- idx_objt_metamap6 on r_objt_metamap (object_id).

The iRODS development group is planning to add all of the new database indexes to subsequent versions of iRODS.

3.1.2. Data

The entire catalog of MODIS Atmosphere data products were registered [13] in this exercise. The modis_Zone contains upwards of 54 million registered files, representing over 630TB of data with over 300 million defined metadata values across the collections. All registered products include the following metadata attributes and related values:

Attribute	**Description**	**Sample Value(s)**
Dataday	Year, month, and Julian day represented by the data.	2006339055527
ESDT	Earth System Data Type (aka product-type)	Atmosphere UARS
OriginalArchiveSet	Original collection under which the product was produced	5
Tile	Numeric tile identifier	51hhhvvv
SatelliteInstrument	Instrument providing the data (MODIS Aqua or MODIS Terra).	Terra MODIS

Additionally, all non-ancillary products contain the following geolocation metadata: OrbitNumber, EastBoundingCoord, NorthBoundingCoord, SouthBoundingCoord, WestBoundingCoord, GungLongitude1, GRingLongitude2, GRingLongitude3, GRingLongitude4, GRingLatitude1, GRingLatitude2, GRingLatitude3, GRingLatitude4, StartTime, and EndTime.

3.1.3. Interfaces

The modrods server functions as our public interface to the modis_Zone. modrods provides web, ftp, and iCommand interfaces to a (currently) limited audience of evaluators/early adopters at Goddard. We also wrote our own additional iCommand, called ilocate, in order to emulate the behavior of the popular GNU locate application.

The standard suite of iCommands is also available for use with the MODIS iRODS collections. The modis_Zone is mounted as a standard unix directory tree via the FUSE interface. An NCFTPd server presides over this FUSE mount, and, as such, provides an ftp-interface into the system. The standard web interface is also available for the MODIS collection via the URL http://modrods.modaps.eosdis.nasa.gov.

3.2. merra_Zone: MERRA Climate Simulation Data

The goal for merra_Zone was to use iRODS to provide access to Modern Era Retrospective-Analysis for Research and Applications (MERRA) model output data. This exercise allowed us to gain experience in an experimental setting using iRODS to manage simulation data products produced by the GMAO.

Retrospective-analyses (or reanalyses) have been a critical tool in studying weather and climate variability for the last 15 years. Reanalyses blend the continuity and breadth of output data of a numerical model with the constraint of vast quantities of observational data. The result is a long-term continuous data record. MERRA was developed to support NASA's Earth science objectives by applying the state-of-the-art GMAO data assimilation system that includes many modern observing systems in a climate framework. The MERRA time period will cover the modern era of remotely sensed data, from 1979 through the present, and the special focus of the atmospheric assimilation will be the hydrological cycle [16].

MERRA data can be accessed from the Goddard Earth Sciences Data and Information Services Center (GES DISC) through a variety of mechanisms, including OPenDAP, FTP, and GDS. The MERRA project supports NASA's Earth science interests by using the NASA global data assimilation system to produce a long-term (1979-present) synthesis that places the current suite of research satellite observations in a climate data context and providing the science and applications communities with state-of-the-art global analyses, with emphasis on improved estimates of the hydrological cycle on a broad range of time scales.

3.2.1. Architecture

The merra_Zone system, which was developed on an NCCS test server, consists primarily of the co-located iRODS application, iRODS/iCAT database, and file system storage. The core iRODS system consists of a single iCAT enabled iRODS server, irodstest:

- Dell PowerEdge r710
- Two quad-core Intel Xeon x5570 processors @ 2.93 GHz
- 24 GB of on-board memory
- 1.2 TB of 15k RPM SAS drives, configured for RAID-5

This instance serves a single zone, merra_Zone, and is made up of a single storage resource, merra_Resc.

PyRods and EmbedPython were used to extend the merra_Zone iRODS installation. The first extension module, known as **iRODS-NCCS**, provides custom micro-services for handling MERRA data. These micro-services and supporting code perform MERRA-specific validation (e.g. checks for HDF4 compliance), internal file metadata extraction, and population of the appropriate metadata iCAT tables. The design, implementation, and documentation of these NCCS-specific policies, mechanisms, and associated metadata produced a road-map for specifying a tailored iRODS archive administrative interface that can be extended and generalized to a variety of client-side interfaces.

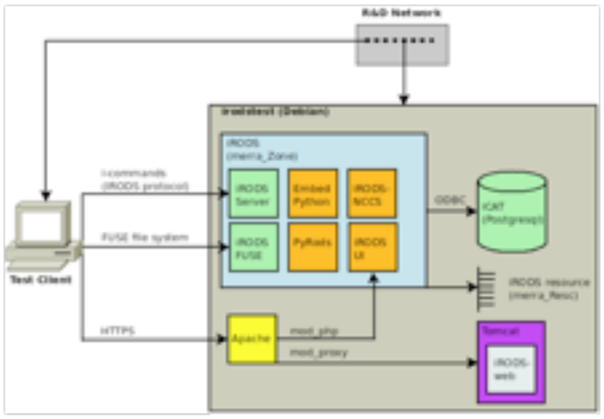
The other set of extended functionality, known as **iRODS-web UI**, was created to enhance the existing auditing capability in iRODS, and create a new reporting mechanism. Auditing in iRODS is rudimentary, consisting of a single database table whose data can only be viewed directly from the database. This table captures an association of data objects, users, actions, and timestamps. Unfortunately, data objects and users can be completely deleted from iRODS during normal operations, leaving historical auditing all but impossible. Moreover, only the Ids of actions are stored in the database, leaving an interested party to compare the Ids against source files to understand their meaning. Finally, the timestamps are stored as strings representing Unix time, making them cumbersome to query.

Our iRODS web enhancements solve these problems by adding new tables and triggers for historical data, lookup tables to decouple meaning from the source code, and views to overcome inadequacies in the field definitions. Furthermore, a new web application (built using Java, Javascript, AJAX) was developed to display audit history (e.g., load time, metadata updates, user reads/downloads) and statistical usage reports (e.g., file size/user, file size/zone, storage rate/time, etc.), which was integrated into the existing iRODS web UI via HTML iframes. The following table identifies all of the software components used within this deployment of iRODS:

Name	**Version**	**Notes**
iRODS	2.3	Core iRODS installation. Includes i-commands.
Extrods	1.1.0.1-beta	Officially provided iRODS web UI.
PHP	5.2.6	Required for iRODS web UI.
Apache web server	2.2.9	Required to serve iRODS web UI.

FUSE library	2.7.4-1.1	Base FUSE library required for iRODS FUSE interface.
Postgresql	8.4.2	Required RDBMS for iCAT.
UnixODBC	2.2.12	Required for iRODS communication to iCAT.
PyRods	2.3	Community provided Python wrapper for iRODS libraries.
EmbedPython	2.1	Community provided iRODS extension that allows for Python based micro-service development.
Python	2.5	Core Python environment, needed for PyRods and EmbedPython.
iRODS-NCCS	0.2	Custom Python based micro-services for MERRA data handling.
iRODS-web	1.0	Java application for viewing iRODS audit history and usage statistics.
Apache Tomcat	6.0.20	Java Servlet container that serves iRODS-web-stats application.
Java Runtime	1.6.0_17	Java runtime, required for iRODS-web-stats.
Ncdump	4.2.5	Library for interacting with HDF files, required for iRODS-nccs
Debian	5	Base OS.

The software and hardware components, including communication mechanisms between components, are shown in the diagram below:



3.2.2. Data

The entire catalog of monthly MERRA products was ingested for the purposes of the prototype merra_Zone system [13]. This resulted in the ingestion of 360 files occupying 47 GB. During the ingestion process, the MERRA data was registered with iRODS and stored on the filesystem. When each file was registered, standard iRODS-based metadata was stored in the iCAT. In addition, MERRA-specific embedded metadata was parsed and stored. The data model provided by iRODS was used to manage the data described thus far. Additional information was also accumulated by extending the iRODS data model. The following metadata attributes were collected on all MERRA products:

Attribute	Description	Sample Value(s)
comment	As required	GEOS-5.2.0
title	Experiment identification: “MERRA”	MERRA reanalysis. GEOS-5.2.0
conventions	file convention	CF-1.0
references	GMAO website address	http://gmao.gsfc.nasa.gov/research/merra/
variables	geophysical quantities	Sea-level pressure, Surface pressure, Surface Geopotential, Geopotential height, Ozone Mixing Ratio
history	CVS tag of this release.	File written by CFIO
contact	contact info	http://gmao.gsfc.nasa.gov/
institution	NASA Global Modeling and Assimilation Office	Global Modeling and Assimilation Office, NASA Goddard Space Flight Center, Greenbelt, MD 20771
dimensions	scale (coordinate) information.	TIME_EOSGRID = 1, YDim_EOSGRID = 144, XDim_EOSGRID = 288, Height_EOSGRID = 42
source	source of data	Global Modeling and Assimilation Office. GEOSops_5_2_0
missing_value	Same as _FillValue. Required for COARDS backwards compatibility.	9.9999999e+14f
hdfeosversion	Version of the HDF-EOS library used to create this file.	HDFEOS_V2.14

To support the iRODS-web application, new constructs were added to the iCAT database schema. All of the constructs are non-intrusive and do not affect core iRODS functionality:

- Lookup table for defining action Id to name mapping:

  ext_audit_actions

- New historical tables that shadow the main iRODS data tables:

  ext_coll_historical **ext_resc_historical**
  ext_data_historical **ext_zone_historical**

- Trigger function for moving data to be deleted into the historical tables:

  tg_ext_audit_delete

- Views for aggregating historical and current data:

  vw_ext_coll **vw_ext_zone**
  vw_ext_data **vw_ext_audit**
  vw_ext_resc

One of the main features of iRODS is the ability to define rules and policies to be applied to data at any step in the life-cycle of that data. In merra_Zone, the following policies were implemented using the iRODS rule engine:

- **Validate the data** – Ensure the file is an HDF file, readable by ncdump and contains appropriate MERRA related header fields and values.
- **Move the file** – If the file was added in the wrong location, move it to the MERRA collection setup within the instance (i.e. /merra_Zone/home/public/merra/<year>)
- **Process the metadata** – Parse the HDF header and associate metadata with the file in iRODS

- **Replicate the file** – If configured for replication, copy the file to another resource for archival purposes.

3.2.3. Interfaces

MERRA users can use iRODS's standard iCommand CLI as well as the iRODS/FUSE CLI to interact with iRODS resources through a POSIX-like representation of a filesystem. FUSE file operations can be intermingled with iRODS's standard iCommands, providing a useful mix of power and familiarity to users.

3.3. yotc_Zone: YOTC Climate Simulation Data

The yotc_Zone system was our second exercise involving model output data. The goal for yotc_Zone was to provide production access to Year of Tropical Convection (YOTC) data. YOTC, a joint activity of the World Climate Research Programme (WCRP) and World Weather Research Programme (WWRP)/ THORPEX, is a year of coordinated observing, modeling, and forecasting focused on organized tropical convection, its prediction, and predictability. The intent is to exploit the vast amounts of existing and emerging observations, the expanding computational resources, and the development of new, high-resolution modeling frameworks [17].

The specific contributions provided by GMAO include both assimilation and forecast products produced every six hours and at higher frequency. The GMAO YOTC data set begins in January 2009 and will extend through the end of the YOTC period. Selected time periods from 2008 may also be available before the completion of the YOTC project.

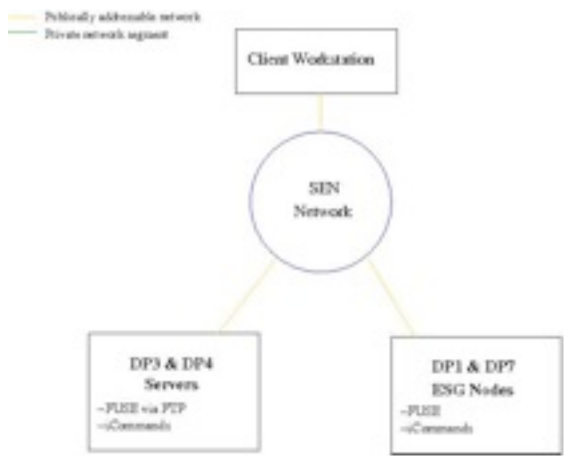
3.3.1. Architecture

The yotc_Zone system, which was deployed on the NCCS Dataportal, consists of the iRODS application and iRODS/ICAT database [18]. The Dataportal implementation use the following hardware:

- 2 Dell PowerEdge R710 Servers
- 4x 2.8GHz Quad-Core Xeon X5570 (Nehalem-class) processors
- 42GB of DDR3 memory clocked at 1333 MHz
- 2x 73GB 15kRPM SAS Hard drives in a RAID1 (mirrored) configuration, for general OS use.
- 1x Dell Powervault RAID Array with 14x 600GB 15kRPM SAS hard drives in a RAID 10 configuration, exclusively for database use.

One of the servers is active, and the other serves as a redundant warm-spare with the capability to take over iCAT services in the event that the active server fails. The diagram below presents the yotc_Zone hardware architecture.

As with the modis_Zone, all iRODS software on the Dataportal has been setup to run as the dedicated system user 'iguy'. This measure establishes a hard de-



gree of privilege separation such that iRODS lacks any write privileges beyond what it absolutely requires.

YOTC data physically resides on our Discover systems, and the NCCS's current security posture only permits YOTC data to be accessed from the Dataportal by way of Discover NFS exports.

As such, our best available implementation method was to scan through the NFS exported filesystems for relevant YOTC data and register any data not yet encompassed within the yotc_Zone. The following scripts were authored in order to carry out this goal:

- **scan_yotc_data**: Scans through the Discover NFS mounts for YOTC data, compiling a list of everything available via NFS. That listing is then compared to already-existing data objects within the yotc_Zone, and finally the script outputs a listing of YOTC files that exist within NFS but not in the yotc_Zone.
- **register_yotc**: Expects a listing of YOTC files to be registered within the yotc_Zone, which is normally provided by the scan_yotc_data script. Data is validated to ensure it is of the proper NetCDF4 type and contains valid headers readable by the ncdump utility. Assuming validity tests are passed, the file is registered into the yotc_Zone, and tagged with relevant metadata attributes, including runID, collectionID, time and geolocation information.

The following table identifies all of the software components used within this deployment of iRODS:

Name	Version	Notes
iRODS	2.3	Core iRODS installation. Includes i-commands.
Extrods	1.1.0.1-beta	Officially provided iRODS web UI.
FUSE library	2.7.2	Base FUSE library required for iRODS FUSE interface.
Postgresql	8.3.11	Required RDBMS for iCAT.
UnixODBC	2.3.0	Required for iRODS communication to iCAT.
PureFTPd	1.0.22	FTP Service backended by FUSE
SuSE Enterprise Linux	11	Base Operating System

3.3.2. Data

The entire catalog of YOTC products was registered [18]. The yotc_Zone consists of approximately 134,000 files representing over 12TB of data with approximately four thousand defined metadata values. Files are all of type NETCDF4.

3.3.3. Interfaces

FUSE provides the primary interface to the yotc_Zone on the Dataportal systems. The yotc_Zone is FUSE mounted on Dataportal nodes dp3 & dp4 to provide an ftp interface into FUSE by way of pure-ftpd. The yotc_Zone is also FUSE mounted on nodes dp1, dp7, and dp15 in order to facilitate publishing YOTC into the Earth System Grid, as described below. The standard suite of iCommands has also been installed on all Dataportal nodes to provide an additional shell interface into the yotc_Zone.

3.4. isds_Zone: ISDS Earth Observational Data

The goal for the Invasive Species Data Service (ISDS), or isds_Zone, was to design, build, and gain experience deploying and using a small-scale, multi-product, application-specific iRODS system. The concept being explored here is using iRODS as means for personal- or laboratory-scale data management, which, in addition to providing iRODS's built-in capabilities for data organization, would convey the added advantage of being able to participate in an extended federation of iRODS resources.

The target application for the isds_Zone is the Invasive Species Forecasting System (ISFS). ISFS is a modeling framework that allows users to load point occurrence field sample data for a plant species of interest and quickly generate habitat suitability maps for geographic regions of management concern, such as a national park, monument, forest, or refuge [19]. Target customers for applications built using ISFS are natural resource managers and decision-makers who have a need for scientifically valid, model-based predictions of the habitat suitability of plant species of management concern. The isds_Zone functions as a local data service for ISFS, managing the input field sample data sets and remote sensing environmental data sets used by ISFS's modeling algorithms.

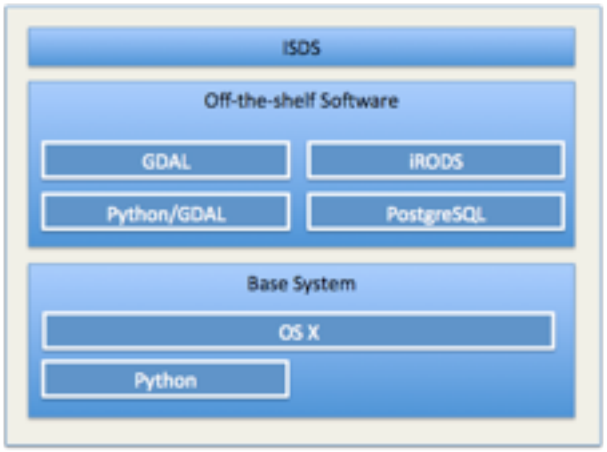
3.4.1. Architecture

The isds_Zone is designed to reside on a standalone computer. In the work described here, the following configuration was used for development and testing:

- iMac
- OS X 10.6.4 Snow Leopard
- 2.8 GHz Intel Core 2 Duo processor
- 4 GB 667 MHz DDR2 SDRAM
- 1 TB disk

The core iRODS architecture consists of a single iCAT enabled iRODS server. This instance serves a single zone, isds_Zone, and is made up of a single storage resource, isds_Resc.

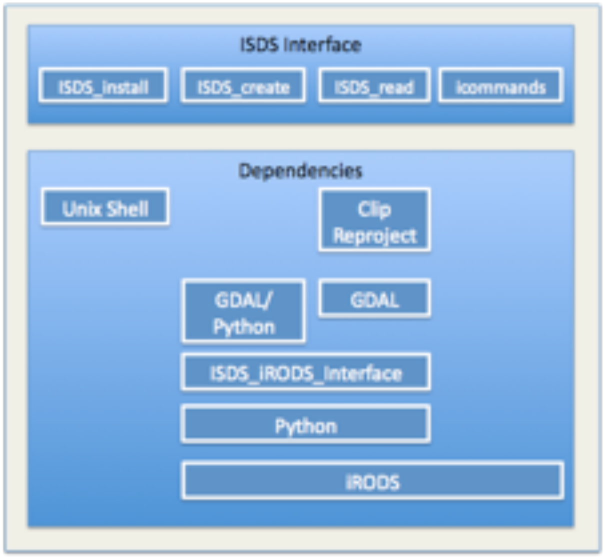
As shown in the figure below, the base system consists of the Mac OS X 10.6.4 (Snow Leopard) operating system, Python 2.6.1, and the Geospatial Data Abstraction Library (GDAL) 1.7.2. GDAL is open-source software to perform geospatial operations on image files. In our implementation, the isds_Zone uses a Snow Leopard binary installation available from KyngChaos that includes Python/GDAL bindings and all GDAL's dependencies: UnixImageIO, PROJ, GEOS, and SQLite3. GDAL must be installed to a base system in /Library/Frameworks, specific to OS X.



isds_Zone was built using iRODS 2.3. The installation requires run-time configuration during the isds_Zone install process. Users are directed through this simple procedure by the ISDS installer. Building iRODS on the base machine requires the Apple Developer Environment.

Python, GDAL functions, and bash scripts were used to extend isds_Zone installation. The collection of program extensions, known as the **ISDS_iRODS_Interface**, provide custom tools for handling MODIS time-series data products of particular importance to the ISFS application. The ISDS_iRODS_Interface provides the fundamental installation storage, retrieval, and processing procedures common to many types of observational data and geospatial applications, as shown below.

ISDS_create extracts metadata from environmental predictors before storing in iRODS. It depends on GDAL/Python bindings to read metadata from predictors, ISDS_iRODS_Interface to store files and metadata in iRODS, and iRODS for data storage. ISDS_read retrieves predictors from iRODS. It depends on clipReproject to trim predictors to a user-defined boundary and reproject them to UTM coordinates. It uses ISDS_iRODS_Interface to retrieve files from iRODS. clipReproject uses GDAL to trim and reproject the predictors.



ISDS_install is a Unix shell script to set up the ISDS environment on a user's base machine. In summary:

- **ISDS_create** extracts metadata from comma separated values (CSV) files and environmental predictors before storing in iRODS.

  Command Summary: ISDS_create [options] <path to input file>
  Options:

  -a "annotation text" associates free text with the input file and stores it as metadata

  -h prints a summary of the ISDS_create interface

- **ISDS_read** retrieves CSV files and predictors from iRODS.

  Command Summary: ISDS_read [options] <*iRODS file name*>
  Options:

  -d *destination_directory* indicates the file system location to which to write the retrieved file

  -d_epsg=*EPSG_code* indicates the user wants the retrieved predictor in a specific projection; ISDS_read will automatically perform the reprojection

  -h prints a summary of the ISDS_read interface

  --ulx=*upper-left_x* indicates the user wants the retrieved predictor clipped to a specific rectangle in ground coordinates; each ordinate, ulx, uly, lrx, lry must be provided; ulx is the westernmost ordinate of the rectangle

  --uly=*upper-left_y* indicates the user wants the retrieved predictor clipped to a specific rectangle in ground coordinates; each ordinate, ulx, uly, lrx, lry must be provided; uly is the northernmost ordinate of the rectangle

  --lrx=*lower-right_x* indicates the user wants the retrieved predictor clipped to a specific rectangle in ground coordinates; each ordinate, ulx, uly, lrx, lry must be provided; lrx is the easternmost ordinate of the rectangle

  --lry=*lower-right_y* indicates the user wants the retrieved predictor clipped to a specific rectangle in ground coordinates; each ordinate, ulx, uly, lrx, lry must be provided; lry is the southernmost ordinate of the rectangle

ISDS_iRODS_Interface wraps the iRODS iCommands iput, iget, and imeta, adding fail-safe error handling and bundling repetitive processes to provide a single-command application programming interface to the higher-level ISDS applications ISDS_create and ISDS_read.

The following table identifies all of the software components used within this deployment of iRODS:

Name	**Version**	**Notes**
ISDS	1	Interface to manage raw data for environmental modeling
iRODS	2.3	Core iRODS installation including i-commands
Postgresql	8.4.2	Required RDBMS for iCAT
GDAL	1.7.2	Required image processing s/w for ISDS
GDAL/Python	1.7.2	Python bindings for GDAL's API
OS X Snow Leopard	10.6.4	Base operating system
Python	2.6.1	Primary ISDS application development language

3.4.2. Data

The isds_Zone system is designed to handle a variety of Earth observational and environmental data sets that can be used as predictors (independent variables) in the regression analyses often used in ecological modeling. However, the most important of these, and the collection that is the focus of this work, is a set of phenology metrics that have been estimated based on temporally smoothed and spatially gap-filled MODIS vegetation indices (VI) over the North American continent. The phenology algorithm has been applied to three MODIS vegetation indices: Leaf Area Index (LAI), Normalized Difference Vegetation Index (NDVI), and Enhanced Vegetation Index (EVI). The spatial coverage of this phenology data is more complete than other remotely sensed data based phenology products and has become particularly important to invasive species habitat suitability modeling [20].

Other important data managed by isds_Zone are bioclimatic variables derived from monthly temperature and rainfall values. These are often used in ecological niche modeling. The bioclimatic variables represent annual trends (e.g., mean annual temperature, annual precipitation) seasonality (e.g., annual range in temperature and precipitation) and extreme or limiting environmental factors (e.g., temperature of the coldest and warmest month, and precipitation of the wet and dry quarters) [21].

Other predictor layers are generated from geographic information system (GIS) data, such as Vegetation Continuous Fields (VCF) obtained from the University of Maryland, and aspect, elevation, and slope constructed from the USGS National Elevation Dataset (NED) and Shuttle Radar Topography Mission (SRTM).

One of the main features of iRODS is the ability to define rules and policies to be applied to data at any step in the life-cycle of that data. In isds_Zone, the following policies were implemented atop iRODS, but may be integrated using the iRODS rule engine:

- **Validate the data** – Ensure the file is a valid field sample or predictor file containing appropriate ISDS related header fields and values.
- **Process the metadata** – Parse field sample and predictor files and associate metadata with the file in iRODS.
- **Read the data** – Retrieve field sample and predictor files, clipping or reprojecting predictors as requested by users.

3.4.3. Interfaces

isds_Zone users can use the iRODS standard iCommand CLI, the iRODS/FUSE CLI, and the iRODS Rich Web Browser to interact with iRODS resources through a POSIX-like representation of a filesystem.

4. Data Management System Extensions

These four systems provide a testbed for exploring the use of iRODS to handle specific challenges that we face in the NCCS. Here we briefly describe three exploratory uses of the testbed that could provide a means of addressing those challenges.

4.1. Observational/Simulation Data Integration

As the title of this paper suggests, integrated access to heterogeneous data is a topic of increasing interest to us, particularly as it applies to coordinated access to observational data and climate model outputs. The iRODS federation mechanism provides one way of accomplishing this integration.

Federation is a feature of iRODS in which separate iRODS zones, can be integrated. When zones 'A' and 'B' are federated, they share otherwise isolated data collections. By eliminating the need to explicitly switch an iRODS client between distinct instances, federation allows perusal or download of data from multiple iRODS systems through a single interface.

For the DMS project, two zones were federated, an ‘observational’ zone consisting of MODIS data and a ‘simulation’ zone consisting of YOTC data. This observational/simulation federation essentially united separate data collections while providing a single consolidated view to the collections.

Each zone in a federation continues to be a separate iRODS instance, administered separately. However, users in multiple zones, if given permission, will be able to access data and metadata in the other zones. No user passwords are exchanged, as each system will, in a secure manner, check with the user's local zone for authentication when the user connects.

The zone name is at the root of each collection name, so for most interactions, the zone is found via the path name, the logical name. For example, /simulation/home/rods/collection1, is in a zone called ‘simulation’. Once configured, the iRODS system will contact the servers in zone ‘simulation’ to access files in /simulation/home/rods/collection.

In the context of the DMS project, our experimental observational/simulation federation consists of the following components:

- YOTC data - Hosted on the Discover cluster and registered in the simulation_Zone
- MODIS data - Hosted on the Dataportal and registered in the observational_Zone
- YOTC iRODS server - iRODS server and iCAT installed in the NCCS Data Portal that contains YOTC registration data and embedded metadata
- MODIS iRODS server - iRODS server and iCAT installed in the NCCS Data Portal that contains MODIS registration data and embedded metadata
- iCommand interface – iRODS command-line interface for federation administration and data management

In this case, federated instances exist locally on NCCS’s Dataportal System. One instance is the yotc_Zone described above, and the other a local modis_Zone, running on the following hardware:

- 1 HP ProLiant Bladeserver 460c, equipped with 2x Quad Core Intel Xeon 2.8 GHz processors
- 8GB DDR3 Memory
- 2x 73GB SAS 10k RPM drives

The basic software configuration for the federated zones is listed below.

Name	Version	Notes
iRODS	2.3	Core iRODS installation. Includes i-commands.
Extrods	1.1.0.1-beta	Officially provided iRODS web UI.
FUSE library	2.7.2	Base FUSE library required for iRODS FUSE interface.
Postgresql	8.3.11	Required RDBMS for iCAT.
UnixODBC	2.3.0	Required for iRODS communication to iCAT.
SuSE Enterprise Linux	11	Base Operating System

In order to establish the federation, the zone administrators use iadmin to define each remote zone and to create remote-zone user entries. For DMS, federated two individual zones within the Dataportal System, the simulation_Zone (dpZone) and observational_Zone (dpZone2):

In the simulation_Zone (YOTC):

```
iguy@dp3:~> iadmin mkzone dpZone2 remote
dp16:1247
```

In the observational_Zone (MODIS):

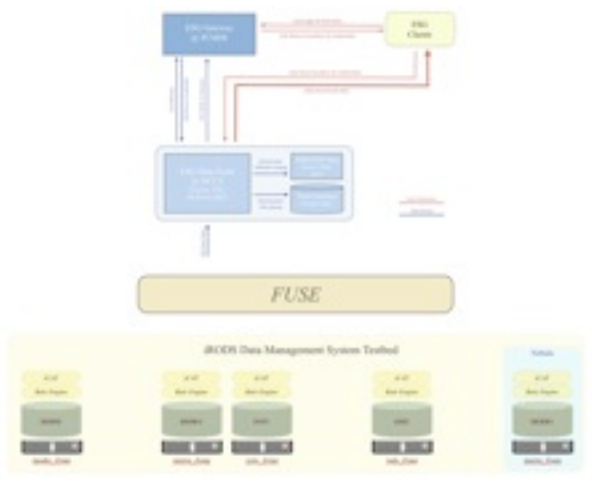
```
iguy@dp16:~> iadmin mkzone dpZone remote
dp3:1247
```

At this point, the users in the observational_Zone can access collections in the simulation_Zone.

4.2. Earth System Grid Integration

Another challenge for us is finding a way to manage NCCS-hosted data products as independent collections in our archive while simultaneously supporting the delivery of those products through community-defined data services. For example, Earth System Grid (ESG) is the system through which AR5 data will be distributed to the IPCC community. ESG integrates supercomputers with large-scale data and analysis servers located at various national labs and research centers located throughout the world [22]. We will publish GMAO and GISS AR5 contributions through an ESG data node located in the NCCS.

As shown below, the iRODS FUSE implementation is one way to abstract archive management from the specific requirements of a data service such as ESG. FUSE is a free Unix kernel module that allows non-privileged users to create their own file systems without editing the kernel code. This is achieved by running the file system code in user space, while the FUSE module provides a "bridge" to the actual kernel interfaces. iRODS FUSE allows normal users and applications to access data stored in iRODS using standard UNIX commands (ls, cp, etc) and system calls (open, read, write, etc).



In the DMS project, we were able to effectively register and publish iRODS-controlled testbed data through the Earth System Grid using iRODS FUSE.

4.3. merra_Zone in the Cloud

Finally, we used the DMS project as an opportunity to build a prototype iRODS data system in the cloud using NASA Cloud Services [23].

NASA Cloud Services provide a progressive, efficient and highly-scalable containerized cloud computing infrastructure. Instead of procuring servers, software, data center space, and network equipment, users can stand up computing storage and virtualization instances in an accessible and affordable pay-as-you go environment. NASA Cloud Services enhance NASA's ability to collaborate with external researchers by providing consistent tool sets and high-speed data connections. NASA Cloud Services are currently being used for education and public outreach, for collaboration and public input, and also for mission support.

NASA Cloud Services are based on Amazon's EC2 cloud model. They provides Infrastructure as a Service (IaaS), which is an aspect of cloud computing that centers on the delivery of platform virtualization as an alternative to traditional data center installations. For this analysis, a base Ubuntu (9.10) image was modified for the IPCC prototype configuration, and was paired with a 60 GB Elastic Block Store (EBS) volume, which is a model of decoupling physical storage volumes from instances in a modular way.

We essentially implemented the merra_Zone system as described above in the NASA cloud. The "irodscloud" host is configured as follows:

- 1 Nebula m1.large cloud instance
- 4 virtualized (KVM/QEMU) Intel processors @ 2.26 GHz
- 4 GB of virtualized physical memory
- 10 GB of virtualized local storage
- 60 GB of virtualized Elastic Block Storage (EBS)

The following table identifies all of the software components used within this deployment of iRODS:

Name	Version	Notes
iRODS	2.3	Core iRODS installation. Includes i-commands.
Extrods	1.1.0.1-beta	Officially provided iRODS web UI.
PHP	5.2.10	Required for iRODS web UI.
Apache web server	2.2.16	Required to serve iRODS web UI.
FUSE library	2.7.4-1.1	Base FUSE library required for iRODS FUSE interface.
Postgresql	8.4.2	Required RDBMS for iCAT.
UnixODBC	2.2.12	Required for iRODS communication to iCAT.
PyRods	2.3	Community provided Python wrapper for iRODS libraries.
EmbedPython	2.1	Community provided iRODS extension that allows for Python based micro-service development.
Python	2.6.4	Core Python environment, needed for PyRods and EmbedPython.
iRODS-NCCS	0.2	Custom Python based micro-services for MERRA data handling.
iRODS-web	1.0	Java application for viewing iRODS audit history and usage statistics.
Apache Tomcat	6.0.20	Java Servlet container that serves iRODS-web-stats application.
Java Runtime	1.6.0_17	Java runtime, required for iRODS-web-stats.
Ncdump	4.1	Library for interacting with HDF files, required for iRODS-nccs
Ubuntu	9.10	Base OS.

5. Discussion

We learned many lessons in the course of the DMS project. Three topics in particular emerged as prominent factors in our assessment of iRODS: rule and micro-service development, namespace virtualization, and iCAT performance at production scale.

5.1. Micro-Service Development

Policies and the mechanisms that implement them are at the heart of any data management framework. iRODS achieves its power and adaptability by using arbitrarily-defined server-side rules and micro-services to specify policies and mechanisms.

Rules are expressed as Event:Condition:Action-set:Recovery-set statements. The actions taken by a rule are performed by micro-services — small, well-defined procedures (generally written in C, in our case written in Python) that perform various tasks relating to the data referenced by the rule. Server-side workflows are created by chaining rules together, and rules themselves can specify rules in a nested hierarchy. Changes to a policy or process can be made at any time with the addition of new rules.

This is a powerful and effective approach. But one of the more interesting issues we dealt with was how to think about iRODS rules in a larger, organizational context. In a perfect world, we would like to have clearly stated policies associated with clear, self-documenting mechanisms — an approach that would streamline the development process; facilitate communication among developers, systems administrators, and users; and enable continuity of understanding and technical support in the face of inevitable staff changes. In our experience, it has not been easy to master rules.

We took a staged approach to ease our entry into the world of rule and micro-service development. As described above, we developed several custom extensions to handle data in the merra_Zone and isds_Zone systems. In both cases, we first scripted operations at a high level using iCommands. Only after prototyping and evaluating operations at this higher level did we attempt coding at the level of rules and micro-services.

The advantage of this approach is that high-level scripting allowed us to combine policies and mechanisms in a readable, easily understood context. It also allowed us to test our implementation using iRODS's robust suite of iCommands. The disadvantage, of course, is that these operations exist outside of the iRODS system and essentially function as custom interfaces, inaccessible to other iRODS clients, such as the Rich Web Browser. They also were inefficient, requiring numerous calls to iput, imeta, etc. to implement each operation.

The mapping of scripted functions to rules and micro-services was at times challenging. The design of the iRODS rule engine has been heavily influenced by logic programming, drawing on concepts such as recursive rule expression and forward rule chaining. It also builds on concepts from fields such as active databases, transactional systems, business rule systems, constraint management systems, service oriented architectures, and program verification. Its current state reflects both the power and complexity of this diverse inheritance, and it is probably fair to say that the syntax and semantics of the iRODS rule engine is continuing to evolve. Taken together, these factors make for a steep learning curve.

If there was a lesson to be learned, it would be that overall risk — perceived or real — and general anxiety about a wholesale organizational commitment to iRODS is elevated by its rule engine complexity. One is unlikely to find experienced people to staff a new development effort, which implies that there may be a fairly long ramp-up as a new team hones is skills with iRODS. Once the investment is made in building a technical team, care must be taken to enable continuity of support should that team leave. Clearly, our efforts to build stable infrastructure around iRODS will advance in direct proportion to the degree to which rule and micro-service development can be simplified and stabilized.

5.2. Namespace Virtualization

Virtualization is a key aspect of iRODS. The iRODS architecture decouples clients from dependencies on physical systems through multiple levels of abstraction, having users and applications interact with data through well-defined sets of logical namespaces. In dealing with this aspect of iRODS, we were reminded about the primacy of the filesystem in this domain: the vast majority of the storage and file manipulations performed by climate researchers relies on classic filesystem methods and constructs. Likewise, existing models, analysis tools, applications, and data services generally set atop POSIX-compliant filesystems.

This makes the iRODS FUSE mechanism particularly important to us, and FUSE filesystem virtualization appears to be the quickest path to integrating iRODS into existing NCCS processes. It provides a way to separate archive management from the idiosyncrasies of data services that need access to iRODS-managed collections, and it provides an interface and way of working that is familiar to our users.

Fortunately, our experiences with modis_Zone, yotc_Zone, and tests involving the Earth System Grid suggest that using FUSE for read-only delivery of data to existing filesystem-based services is fast enough to accommodate publication and distribution needs. Writing data to iRODS-managed collections through FUSE, however, is another matter. The write performance of user space filesystems is inherently and notoriously slow. Regardless of how effective FUSE is in the short term, the full effectiveness of iRODS will clearly require other mechanisms. We will need to help our customers become familiar with more direct access approaches to iRODS collections, such as the iCommands and Rich Web Browser, and perhaps even create interfaces tailored to their specific needs.

5.3. iCAT Performance and Optimization

Another crucial element of an iRODS system is its metadata catalog, called the iCAT. iCATs store descriptive state information about the data objects in iRODS collections in an underlying DBMS, in our case, PostgreSQL. Since virtually every key interaction with an iRODS system involves the iCAT, understand its behavior and performance is of interest.

Our intent with modis_Zone was to gain iCAT experience with a large, production data collection. There are 54 million files in modis_Zone associated with over 300 million metadata values. Files were registered across 20 storage nodes at an initial rate of about 25 files per second (a little over two million files per day). After a million files, the process slowed to about one file per second. We were able to restore performance by creating multicolumn b-tree indexes for the iCAT.

After the registration process was complete, we experienced significant performance problems on imeta searches, which could take as long as fifteen to twenty minutes over the entire collection. Here again, performance was improved by adding indexes. The one column indexes described above enabled imeta searches taking two seconds or less. These indexes have been added to the base iRODS software suite as of Version 2.4.1.

The only remaining size-related frustration involves the Rich Web Browser: it takes a long time to populate the pull-down menu of attribute names, making it unusable for large collections as currently configured. The Browser issue notwithstanding, our impression is that the iRODS iCAT can deliver reasonable performance on the types of collections that we will manage, but that obtaining that performance will require straight-forward optimizations to tailor the iCAT to the local context.

6. Conclusions

The figure below illustrates our long-term goal for data services in the NCCS. Ultimately, we would like to be able to provide full information lifecycle management to diverse collections within a uniform, coordinated environment. This environment — a production NCCS Data Management System — would be accessible through direct interfaces, would provide storage for analysis and visualization applications, and would be a platform on which various data servers, tailored to the needs of a diverse and growing customer base, could set.

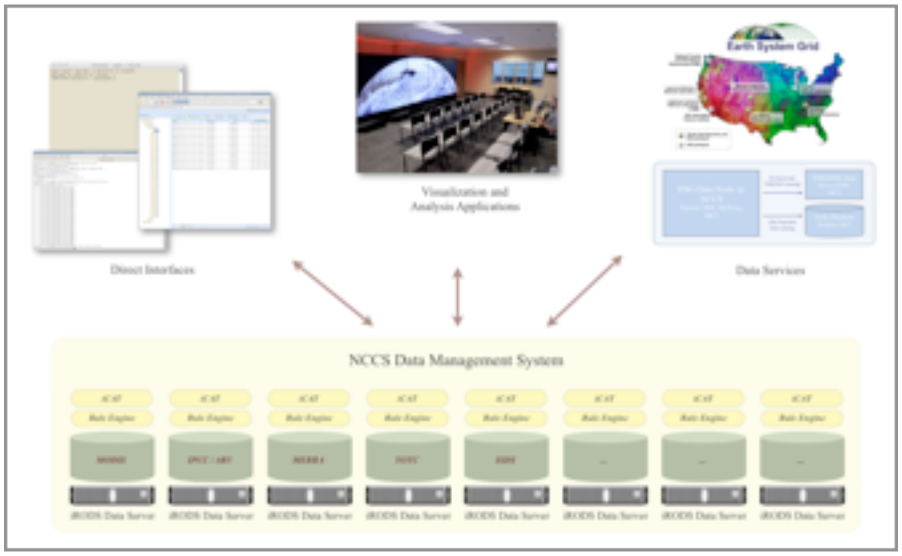
We have been impressed with the potential that iRODS offers for building an integrated capability such as this. This year, we will use the arrival of IPCC AR5 data as an opportunity to use iRODS in an operational setting. Specifically, we plan to develop the policies, mechanisms, rules, and micro-services to manage IPCC data in the NCCS archive and use the iRODS FUSE mechanism to accommodate Earth System Grid publication of GMAO and GISS AR5 data products.

Controlling risk as we adopt this technology is still a major concern, and this staged, incremental approach is one element of our mitigation strategy. Simplifying the approach to rule and micro-service development and support also is a high priority for us, and we will be looking for opportunities to work with the iRODS development team and the iRODS community to foster greater use of Python, for example, for scientific applications such as ours.

The development of an archive administrators toolkit, or interface, tailored to the needs of the NCCS (and perhaps more generally to other climate data centers) also may become an element of our work program. And time permitting, further consideration of iRODS in the context of cloud computing will undoubtedly be part of our future work.

Acknowledgments

We thank our colleagues in the NCCS for their many contributions to this effort. Dan Duffy, Tom Schardt, Ellen Salmon, Lynn Parnell, Phil Webster and Ed Masuoka were instrumental in making the DMS project possible and provided many helpful insights. Yingshuo Shen, Tom Maxwell, and Laura Carriere offered important input about Earth System Grid. Thanks also to the iRODS development team at the University of North Carolina at Chapel Hill for their indispensable technical advice. This work has been funded in part by the American Recovery and Reinvestment Act of 2009.



References

[1] NASA Center for Climate Simulation (NCCS). htttp://www.nccs.nasa.

[2] NASA Science Mission Directorate. 2010. *Responding to the Challenge of Climate and Environmental Change: NASA's Plan for a Climate-Centric Architecture for Earth Observations and Applications from Space*. http://science.nasa.gov/media/medialibrary/2010/07/01/Climate_Architecture_Final.pdf.

[3] Allen, R.B. 2010. *Management and Analysis of Large Scientific Data Sets*. http://ww.grids.ac.uk/NWGrid/LargeData.

[4] Easterbrook, S. 2010. *Climate Change: A Grand Software Challenge*. http://www.easterbrook.ca/steve/?p=1858.

[5] NASA Global Modeling and Assimilation Office (GMAO). http://gmao.gsfc.nasa.gov.

[6] NASA Goddard Institute for Space Studies (GISS). http://www.giss.nasa.gov.

[7] Intergovernmental Panel on Climate Change (IPCC). http://www.ipcc.ch.

[8] Teixeira, J., Waliser, D., Crichton, D., Ferraro, R., Hyon, J., Gleckler, P., Taylor, K., Williams, D., Lee, T., Kaye, J., Maiden, M., Berrick, S. 2010. *NASA Observations for the IPCC*. http://www.clivar.org/organization/wgcm/wgcm-14/talks/061010/NASA_obs.pdf.

[9] NASA Science Mission Directorate. 2008. *Computational Modeling Capabilities Workshop Final Report*. http://www.hec.nasa.gov/workshop08/final_report.pdf.

[10] Integrated Rule-Oriented Data System (iRODS). http://www.irods.org.

[11] Rajasekar, A., Wan, M., Schroeder, W., and Moore, R.W. 2009. *Policy virtualization using Rule-based Data Grids*. http://www.irods.org/pubs/DICE_Rule-based-grids.pdf.

[12] Moore, R.W., Rajasekar, A., and Marciano, R. (Eds.). 2010. *Proceedings of the iRODS User Group Meeting 2010: Policy-Based Data Management, Sharing, and Preservation*, (March 24-26, Chapel Hill, NC), 77 pp.

[13] Moderate Resolution Imaging Spectroradiometer (MODIS). http://modis.gsfc.nasa.gov.

[14] MODIS Atmosphere Data Products (MODAPS). http://modis-atmos.gsfc.nasa.gov.

[15] Goddard Earth Sciences Data and Information Services Center (GES DISC). http://daac.gsfc.nasa.gov.

[16] Modern Era Retrospective-Analysis for Research and Applications (MERRA). http://gmao.gsfc.nasa.gov/research/merra/intro.php.

[17] Year of Tropical Convection (YOTC). http://gmao.gsfc.nasa.gov/projects/yotc.

[18] NCCS Dataportal. http://www.nccs.nasa.gov/dataportal_front.html.

[19] Schnase, J.L., Most, N., Gill, R., and Ma, P. 2009. The Invasive Species Forecasting System: A workflow-oriented decision support framework for managing biological invasions. In: *Proceedings of the 17th International Conference on Geoinformatics (Geoinformatics 2009)*, (August 12-14, Fairfax, VA), 14 pp.

[20] Bin, T., Morisette, J.T., Wolfe, R.E., Feng, G., Edere, G.A., Nightingale, J., and Pedelty, J.A. 2008. Vegetation phenology metrics derived from temporally smoothed and gap-filled MODIS data. In: *Proceedings of the Geoscience and Remote Sensing Symposium (IGARSS 2008)*, (July 7-11, Boston, MA), pp. III-593 - III-596.

[21] Bioclamtic Data . http://www.worldclim.org/bioclim.

[22] Earth System Grid (ESG). http://www.earthsystemgrid.org/about/overview.htm.

[23] NASA Cloud Services. http://nebula.nasa.gov.