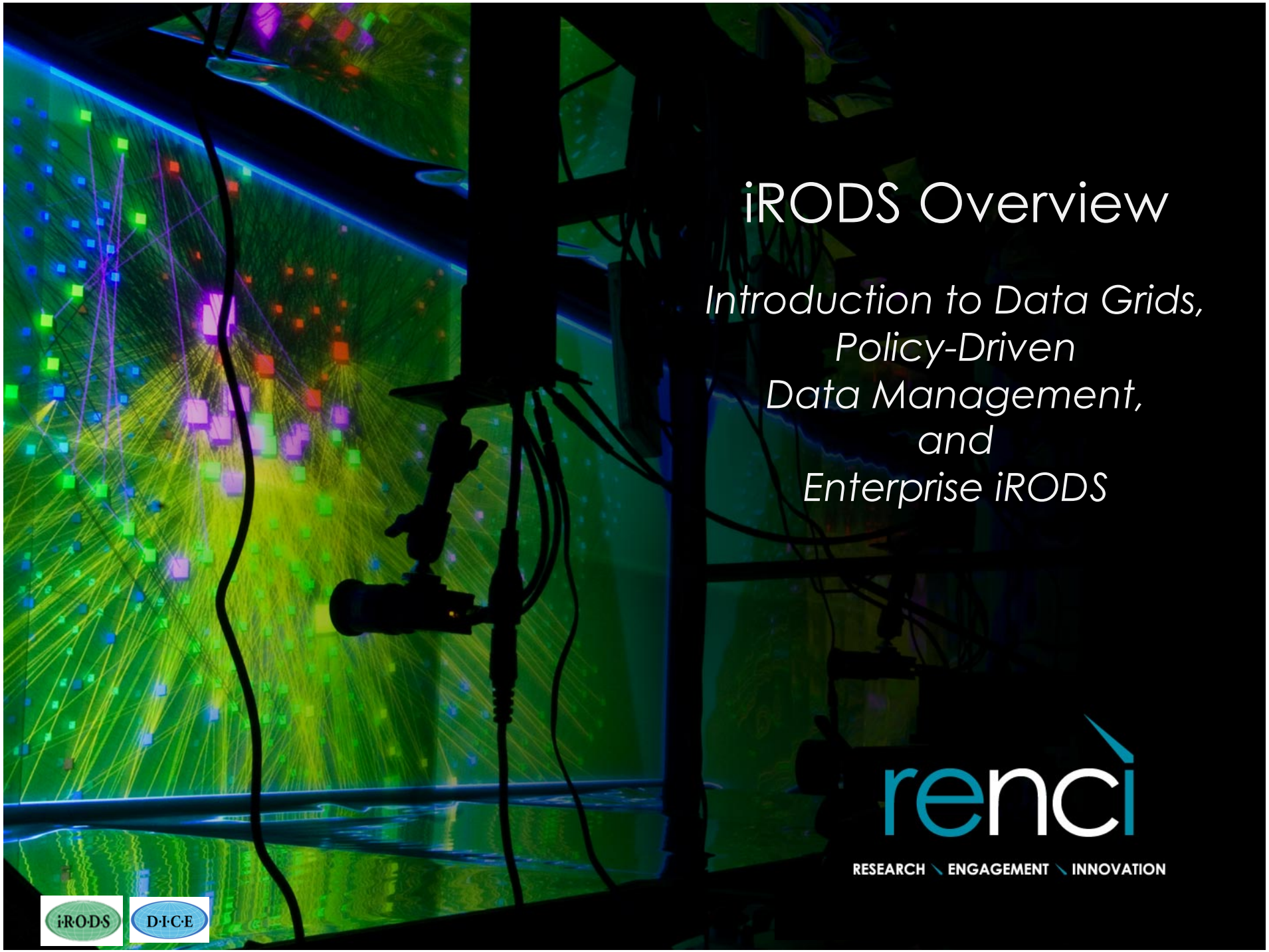


# iRODS Overview

*Introduction to Data Grids, Policy-Driven Data Management, and Enterprise iRODS*

renci

RESEARCH \ ENGAGEMENT \ INNOVATION

iRODS

D·I·C·E

# Renaissance Computing Institute (*RENCI*)

- A research unit of UNC Chapel Hill
- Directed by Stan Ahalt, formerly from the Ohio Supercomputer Center
- State-supported
- Governed by the Triangle universities:
  - UNC Chapel Hill
  - NC State University (Raleigh)
  - Duke University (Durham)

# Data Intensive Cyber Environments (*DICE Group*)

- Directed by Reagan Moore
  - Joint appointment at UNC: at RENCI and at the School of Information and Library Science (SILS)
- Developed SRB, the Storage Resource Broker
- Began at SDSC, the San Diego Supercomputer Center
- Most of the group migrated to UNC Chapel Hill in 2008-2009
  - The group is bi-coastal: DICE-UNC, DICE-UCSD
- Released iRODS, the integrated Rule-Oriented Data System, in 2009

# iRODS Evolution

- Based on decade-long SRB development experience for managing distributed data
- Community-driven
- iRODS picked up where SRB left off
- Modular, extensible, customizable
- Open source (BSD license)
- Supported at UNC by DICE and RENCI

# iRODS

I. Data grid middleware

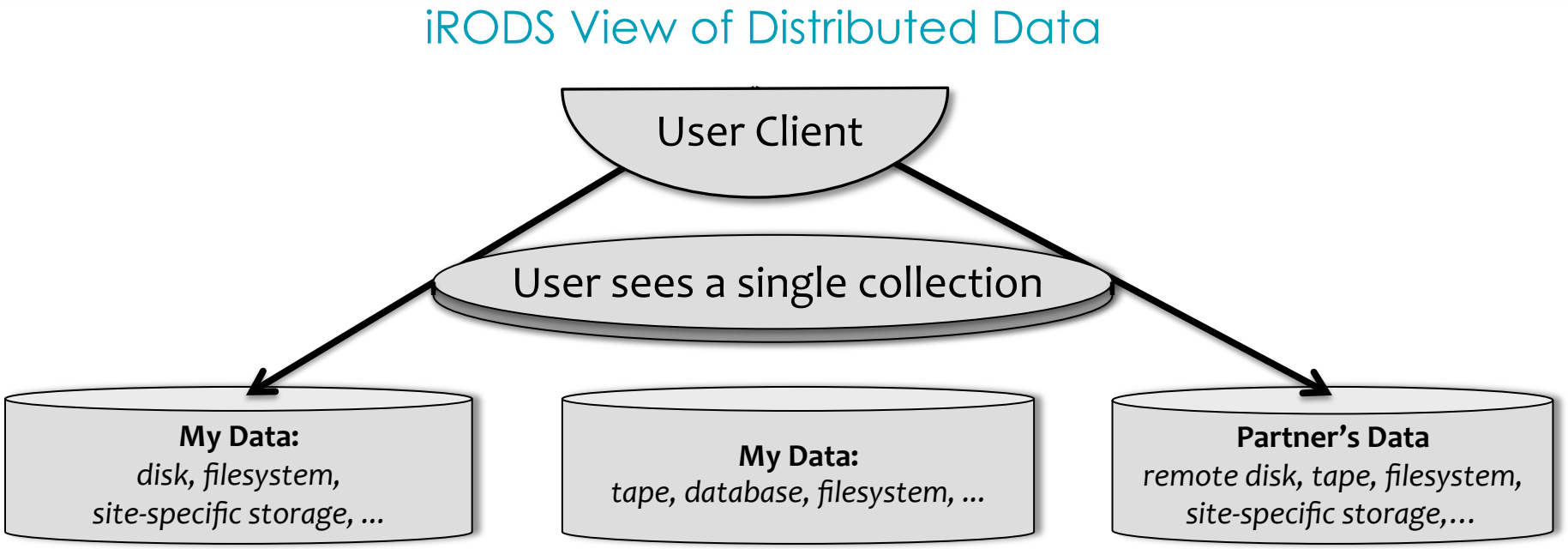
II. Data management infrastructure

III. A framework for procedural implementation of data management policy (policy-driven data management)

iRODS is all these.

# I. iRODS as Data Grid Middleware

# iRODS Unified Virtual Collection



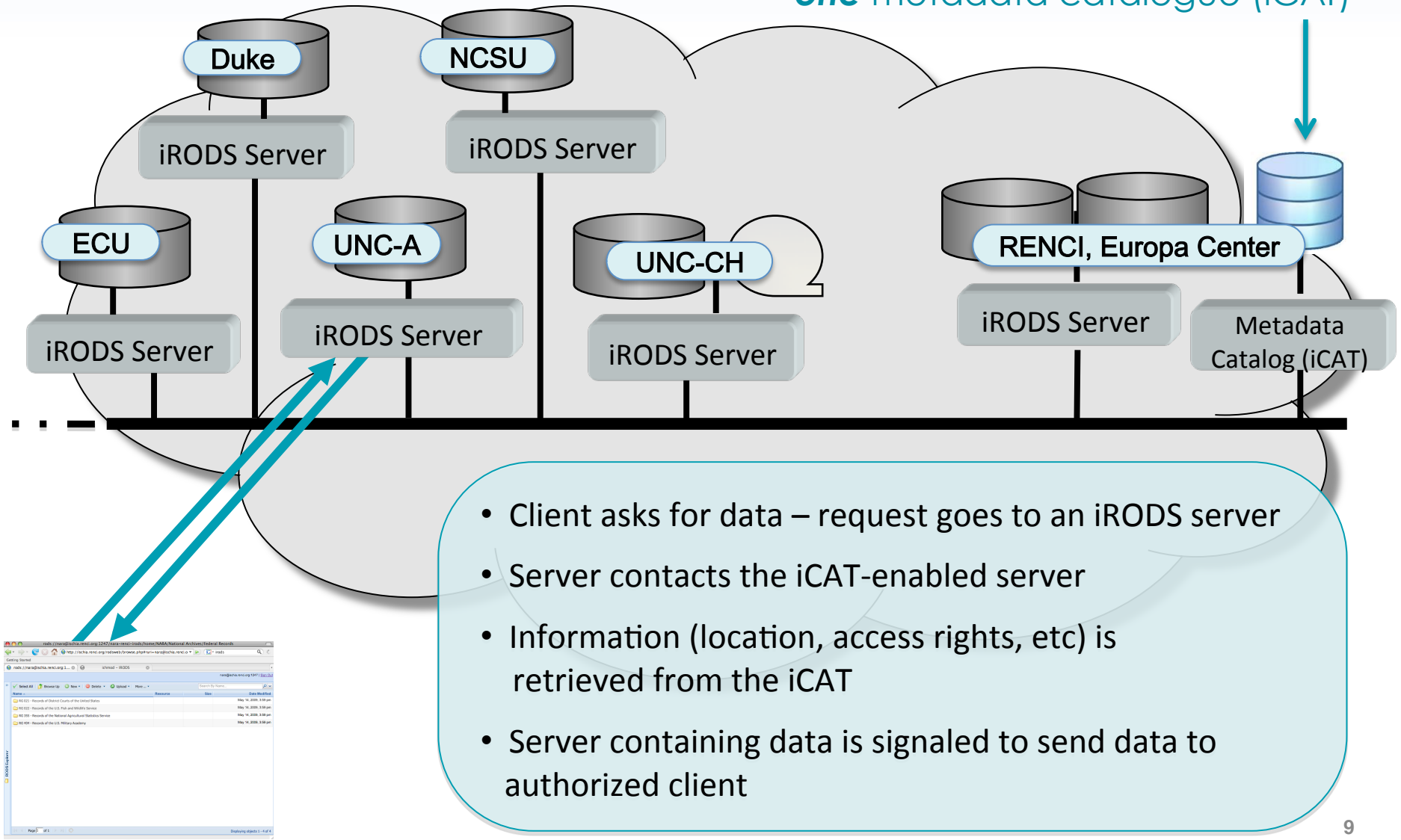

- iRODS installs over heterogeneous data resources
- Users can share & manage distributed data as a single collection

# iRODS as a Data Grid

- Sharing data across:
  - geographic and institutional boundaries
  - heterogeneous resources (hardware/software)
- Virtual (logical) collections of distributed data
- Global name spaces
  - data: files and collections
  - users: single sign on
  - storage: virtual resources
- Metadata catalogue (iCAT) manages mappings between logical and physical name spaces
- Beyond a single-site repository model

# A RENCI Data Grid

A complete data grid (**zone**) has ***one*** metadata catalogue (iCAT)





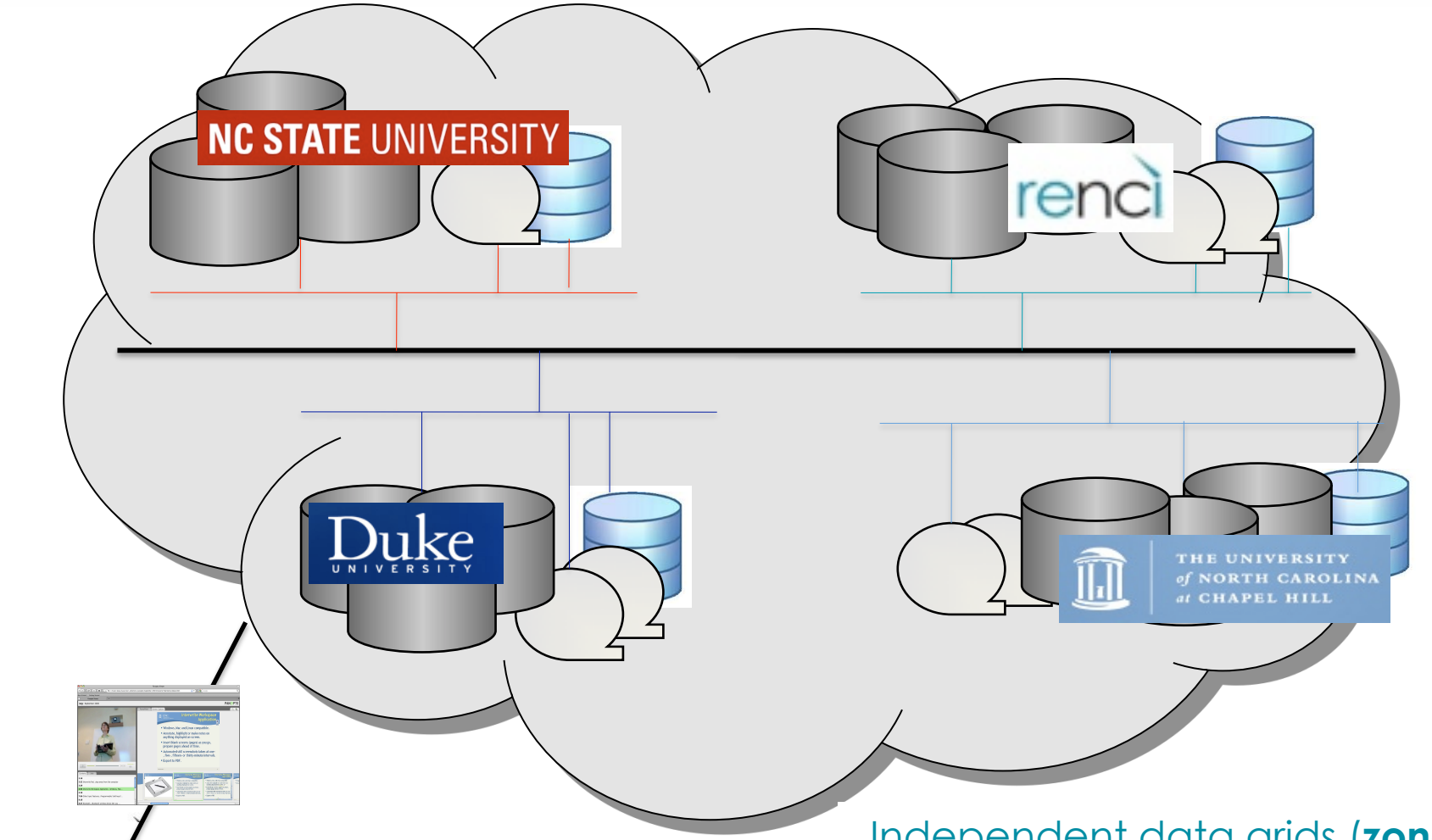

- Client asks for data – request goes to an iRODS server
- Server contacts the iCAT-enabled server
- Information (location, access rights, etc) is retrieved from the iCAT
- Server containing data is signaled to send data to authorized client



iRODS D·I·C·E

# TUCASI Infrastructure Project (TIP)

## Federated Data Grids




Independent data grids (**zones**), each with its own **iCAT**, can be *federated*

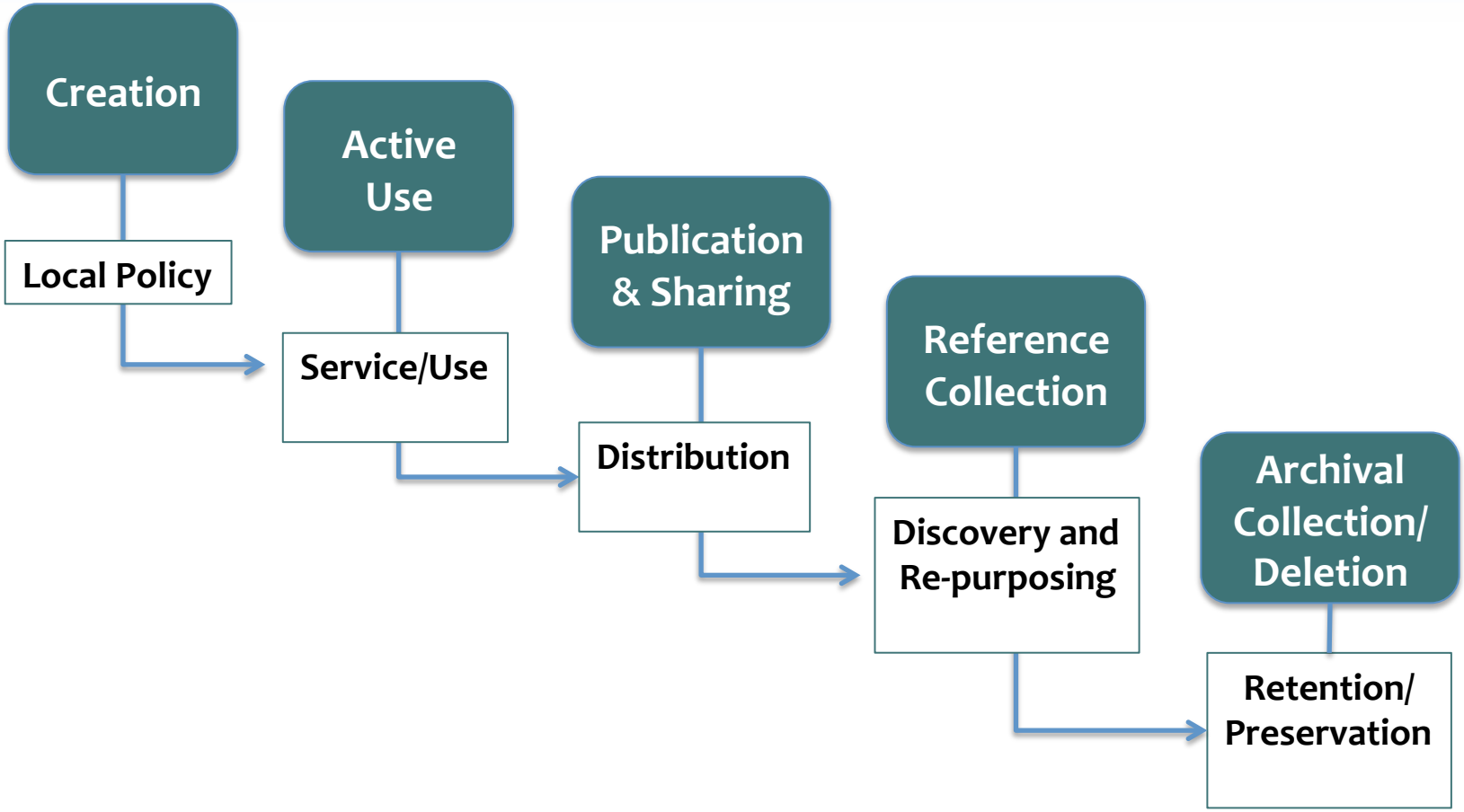
# II. iRODS for Data Management

# Issues in Data Management

## Organization and Usage

- Distributed data/virtual collections
- Distributed access to data (groups); data sharing (remote access and permissions/protection)
- Publishing (general distribution)
- Back-ups and replicas
- Metadata collection and tagging
- Evolution in usage model (life cycle)

# Data Life Cycle




Usage evolution across the stages of the data life cycle.

# More Issues in Data Management

Requirements on Data Infrastructure

- Data integrity, authenticity/verification, provenance
- Discoverability (metadata management and query support)
- Audit tracking/accounting
- Reanalysis, reproducibility
- Interfaces to the data
- Data services (derived products, new formats,…)

# Further Considerations
## (from the archivists)

- Ingestion
  - How will data be submitted?
  - Virus checks
  - Data types
  - Necessary metadata
- Sharing
  - Access rights
  - Copyright and intellectual property terms
- Curation
  - Setting policy requirements into action – implementation
  - Managing remote data
- Preservation
  - Long-term: life span, archival policy
  - Metadata to support policy

# iRODS for Data Management

iRODS infrastructure supports these and other capabilities for data management.

Need an overall data management plan in order to move to implementation.

Clear statement of data management plan = data policy

iRODS can be used to procedurally implement that data policy.

# III. iRODS for Policy-Driven Data Management

# What is Policy-Driven Data Management?

*Policy determines when management procedures are run*

- Define a data policy
- Identify the management steps to carry out the policy
- Define computer procedures for implementation
- Trigger the procedures when policy requires
  - Events in the data grid such as
    - putting, getting, and replicating collections or files;
    - creating users, resources, groups

    can trigger procedures.
  - State of the data grid can trigger procedures, such as
    - when a given time period has elapsed
    - whenever a file of prescribed format is ingested
    - when data reaches a prescribed age

# Policy-Oriented Data Infrastructure

- Implement management policies
  - Each community defines its own policies
- Manage administrative tasks
  - Data administrator or data proprietor (not system administrator) manages data collections
- Assessment criteria for checking policy compliance
  - Point-in-time: queries on metadata catalog
  - Compliance over time: queries on audit tables

# Policy-Based Data Environments

- Assembled/Distributed Collections
- Properties - attributes that ensure the *purpose* of the collections
- Policies - methodologies for enforcing desired *properties*
- Procedures - functions that implement the *policies*
  *(implemented as computer actionable rules/workflows)*
- Persistent state information - results of applying the *procedures*
  *(contained in system metadata)*
- Assessment criteria - validation that *state information* conforms to the desired *purpose*
  *(mapped to periodically executed policies)*

# Additional iRODS Design Goals

Abstract out the data management

- Policy-based data management
- Separate policy enforcement from storage administration
- Policy follows data around the grid: collection management independent of remote storage locations

Scalability and extensibility

- Enable versioning of policies and procedures and data [planned]
- Support differentiated services (storage, metadata, messaging, workflow, scheduling)
- Pluggable microservice & driver modules [planned]

# iRODS Policy Implementation

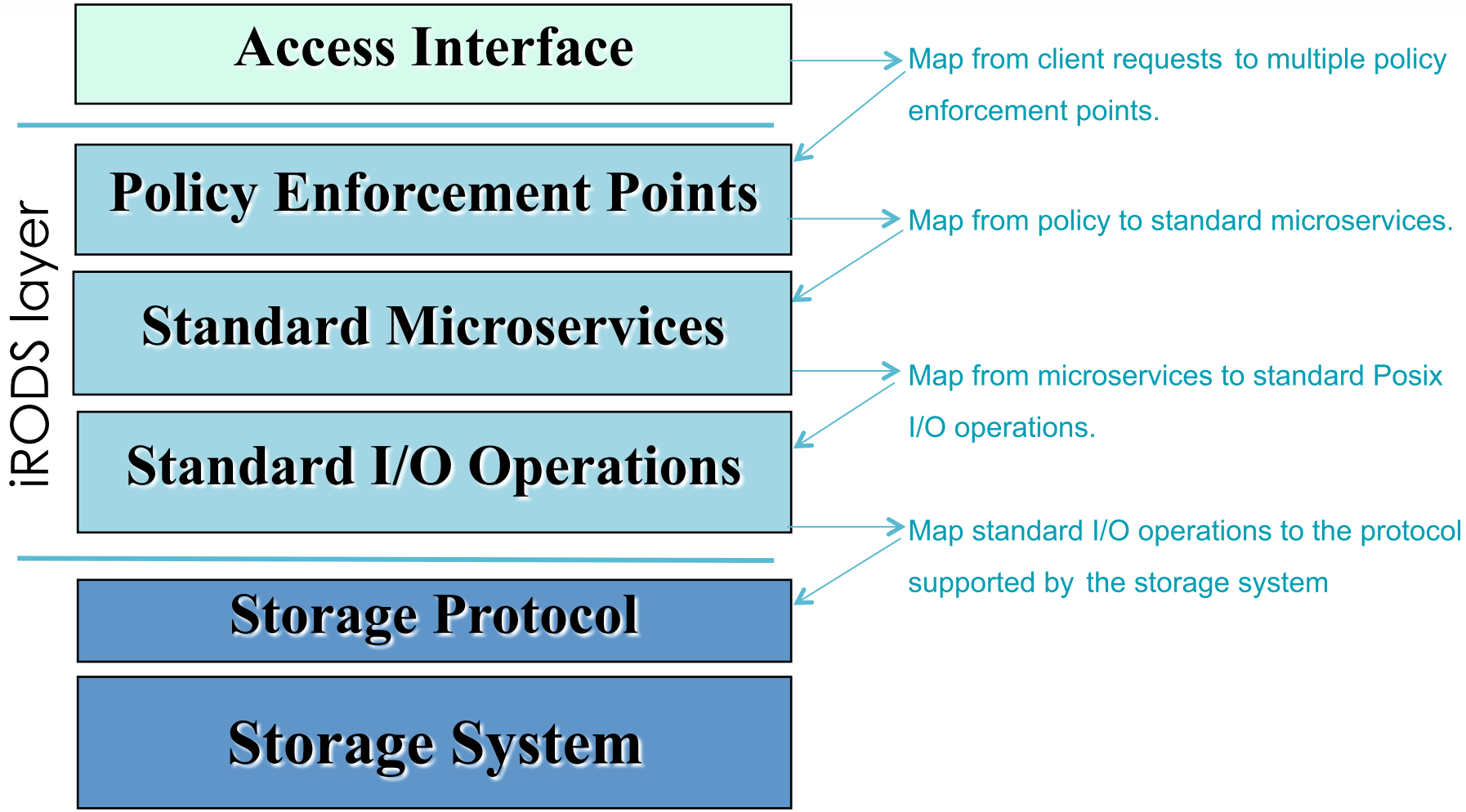
## *Microservices and Rules*

- Microservice – the functional unit of work (C programs)
- Rules – workflows of microservices (and rules)
- Provide server-side (data-side) services
- *Event-triggered rule execution*

# Microservices

- Modular
- New libraries of microservices can be developed without touching the core code
- Allow customization and extension of the data grid for community-specific policy

# iRODS Data Virtualization



# Server Functions

- Peer-to-peer architecture
  - iCAT-enabled server (IES) interacts with metadata catalog
  - authorized requests are forwarded to server where data reside
- Translate from client action to storage protocol
- Authenticate and authorize requested operations
- Implement/enforce policies from local rule engine
- Manage execution of processes at the storage location

# Highly Controlled Environment

- All accesses are authenticated
  - GSI / Kerberos / Challenge-response / Shibboleth
  - Additional mechanisms in development (PAM)
- All operations are authorized
  - ACLs on files, storage
  - Constraints on each rule (only authorized users can run them)
- Local rule base controls interactions with local storage

# Some Management Capabilities

- Replication
- Registration of files into the data grid
- Synchronization of remote directory
- Managed file transport (iDrop)
- Automated metadata extraction
- Queries on metadata, tags
- Server-side workflows (loop over result sets)
- Parallel I/O streams & RBUDP (Reliable Blast UDP) transport

renci

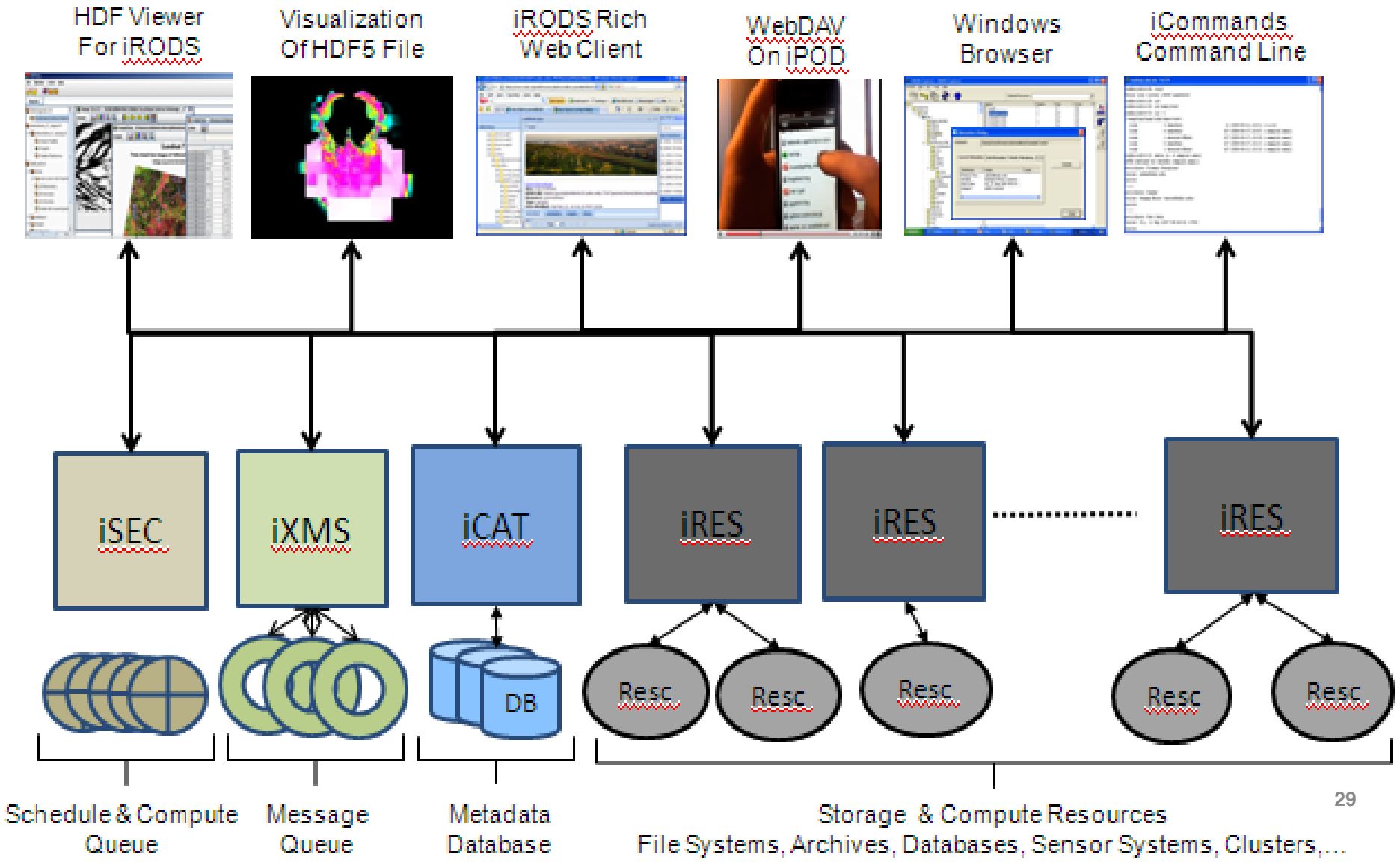
i·R·O·D·S D·I·C·E

# iRODS Extensible Infrastructure

- Some extensions handled by user groups
  - Clients
  - Policies
  - Procedures

- iRODS (extensible) core infrastructure:
  - Network transport
  - Authentication/Authorization
  - Distributed storage access
  - Metadata management
  - Messaging
  - Rule engine

# An iRODS Overview



# iRODS Version 3.0

- Released autumn 2011
- New features
  - New rule engine
    - Improved, cleaner syntax
    - Strong parameter typing
    - Optimized performance (can run thousands of rules)
    - Expanded math operators
  - Extended rule language
    - Rule versioning
    - Rule debugging
    - Distributed rule-based management
  - Symbolic links to external systems (based on microservices)
  - Restart for transport of large files
  - Improved Java interface: Jargon
- Pending features
  - Windows native client port, Windows server support
  - Dropbox interface (iDrop)

# iRODS Books

- The integrated Rule-Oriented Data System (iRODS) Microservice Workbook
  - Available from Amazon: http://www.amazon.com/dp/1466469129
  - Describes new rule language
  - Lists examples for using the current microservices
    - Input / output parameters defined
    - Executable rule provided for each microservice
    - Testing of microservice execution or
    - Illustration of use with other microservices or
    - Demonstration of use in a management policy

- iRODS Primer: integrated Rule-Oriented Data System (Synthesis Lectures on Information Concepts, Retrieval, and Services)
  - From Amazon at http://www.amazon.com/dp/1608453332

# Some Applications

- Astronomy – NOAO, CyberSKA, LSST
- High Energy Physics – BaBar, KEK
- Earth Systems – NASA (MODIS data set)
- Australian Research Collaboration Service (ARCS)
- BeSTGRID – coordinating New Zealand research organizations (public and private)
- Genomics – Broad Institute (MIT, Harvard), Sanger Institute (UK)
- Carolina Digital Repository
- Texas Digital Libraries
- Seismology – Southern California Earthquake Center
- Bibliothèque Nationale de France

# Yearly iRODS User Meetings

- The 2011 Meeting
  - February 2011
  - Chapel Hill, NC
  - Organized by the DICE group, hosted by RENCI

- The 2012 Meeting
  - March 1, 2
  - Tucson, Arizona
  - Organized by the DICE group, hosted by the iPlant consortium
  - National and international participation by many in the iRODS user community

- The 2013 Meeting (tentatively)
  - Feb - March 2013
  - University of Cologne

# Enterprise iRODS: E-iRODS

- RENCI's and UNC's long-term support for iRODS
- Target new funding models for sustainability – move beyond traditional public research funds
- Beta 2 release based on iRODS 3.0 out in June 2012 at e-irods.org
- Organization:
  - Research code (DICE) released about every 4 months
  - Enterprise code (RENCI) released about every 18 months
  - Service agreements and consulting negotiable
- E-iRODS Consortium to fund Enterprise support infrastructure

# E-iRODS 3.0

- Initial release based on iRODS 3.0
  - tracks community code, with a delay
- Hardened binary release of iRODS
  - Passes continuous integration with back-ported bug fixes from community trunk
  - Packaging: initially RPM and DEB
- Certification
- Enhancement of modularity
  - Pluggable microservices
  - Pluggable drivers
- Documentation

# Certification

- 100% test coverage of server-side APIs across selected platforms and topologies: n-way testing across all combinations

- Packages released, as of June 2012
  - DEB (Debian, Ubuntu)
  - DMG (MacOSX) – Unix client (icommands)
  - RPM (RHEL, CentOS, Fedora, SuSE)

- Planned:
  - Solaris
  - Windows (MSI)
  - MacOSX (servers)

- Topologies
  - Single zone: iCAT server + 2 non-iCAT servers
  - Federation: two single zones

# Proposed Support

*Options for Negotiation*

- Tutorials
  - User and admin tutorials
  - On-site hands-on or web conferencing
- Technology preview
  - Tier 2,3 helpdesk response to usage problems: iRODS, E-iRODS
- Production support
  - Bug fixes and problem closure for E-iRODS supported components on supported platforms
- Development support
  - Community or proprietary feature development

# RENCI iRODS Automated Testing

Track testing by a functional Testing Matrix – dimensions represent variables which affect the behavior of data grid:

Platform, Configuration, Database, System Features

Automate an exhaustive walk of the Testing Matrix

Libraries of test scripts for System Features

Regular testing driven by Hudson via a Celery Framework which runs on a Virtual Distributed testing grid

Also useful for non-functional testing: Load Testing, Scalability

# RENCI Collaborative Development and Test Environment

Git – distributed revision control system

GForge – project management system

- hosting & version control
- bug-tracking
- messaging

Hudson/Jenkins – Continuous Integration environment: incremental quality control

Nexus – Maven repository that tracks dependencies and bundles for check-out (Java)

# Continuous Integration

Automated via Hudson (moving to Jenkins)

A risk reduction technique

Push code frequently to the repository

Build & test for each new commit in order to catch defects as early as possible

Automated CI removes a level of burden from developers and provides constant insight to the state of the project

# Open Source Software for the Test Environment

- git
- python
  - celery
  - nose
- erlang
  - rabbitmq
- javascript
  - node.js
- bash

*Developed at RENCI:*

- gridbundle
  - schema.json
  - validator.js
- deploy_gridbundle.py
- assertiCmd/ assertiCmdFail

# Code Hardening

Defensive Programming – anticipate errors and design to avoid them or identify them immediately when they occur

Leverage Static Analysis Tools – audit code regularly for common programming errors, potential vulnerabilities, and security concerns; enhance *code coverage*

Peer review of code to catch errors missed by static analysis as well as potential design issues and opportunities for refactoring

Refactor code, leveraging industry "best practices" for security, extensibility, & maintainability

# Documentation & Support

Create iRODS Server installation packages for supported platforms: RPM, DEB, MSI, etc

Polish the installation procedure & scripts for iRODS

Add support for remote iCAT configuration, automated MySQL installation, etc.

Work towards a comprehensive offline Administration Guide

Begin a Configuration Cookbook: rule configurations for well-known use cases

# E-iRODS Consortium

## *Summary*

- Membership dues will fund basic E-iRODS development
- E-iRODS remains completely open source (binary and source code), with a one-release lag to non-members
- Membership levels
  - Patron Sponsor:
    - Access to release roadmap
    - Privileged access to (paid) consulting and technical support
  - Major Sponsor… patron sponsor plus:
    - Voting rights to release roadmap
    - Non-voting seat on governing board
  - Sustaining… major sponsor plus:
    - Voting seat on governing board
  - Major Sustaining sustaining plus:
    - Enhanced voting rights on governing board