Evolution of Data Grid Technology

Arcot Rajasekar, Michael Y.-K. Wan, Reagan W. Moore, and Hao Xu

***Abstract*—The Data Intensive Computing Environments group has been developing data grid technology for twenty years. Two generations of technology were created, the Storage Resource Broker - SRB (1994-2006) and the integrated Rule Oriented Data System - iRODS (2004-2016). Both products represented pioneering technology in distributed data management and were widely applied by communities interested in not only publishing data, but also sharing and preserving data. Applications included national digital libraries, national data grids, national archives, and international collaborations. The success of the software was strongly driven by basic concepts that still represent the state-of-the-art for data management systems. These concepts include policy-based data management, virtualization, collection life cycle, and federation. The development, evolution, and application of these concepts in data grids, digital libraries and archives are reviewed in this paper.**

***Index Terms*—Collections, Data grids, Federation, Policy-based, Virtualization**

I. Introduction

Data grids, digital libraries, and archives implement basic data management functionality related to ingestion, arrangement, description, storage, and access. The characterization of data management functionality can be done in terms of these services, or through a description of the architecture that supports the services, or through the basic concepts that enable the implementation of the operations. Variants of the basic data management operations are present in data sharing environments (data grids), data publication systems (digital libraries), data preservation systems (archives), and data processing systems. A significant research question has been whether it is possible to design generic data management infrastructure that is capable of supporting not only the digital library operations, but also data sharing and preservation operations. The expectation is that generic versions of data management functionality can be defined and implemented that support publication, sharing, and preservation of data.

The implementation of two successful data management software systems – the Storage Resource Broker (SRB) [2] and then the integrated Rule Oriented Data System (iRODS) [13], represents an example of a software development life cycle for data management systems. User requirements from the academic science community drove the implementation of data management technology, and the evolution of data grid capabilities from simple data management to information and knowledge management. The SRB and iRODS systems pioneered significant conceptual ideas and technologies in large-scale data management and indeed added multiple terms to the ever changing vocabulary of the field. The current emergence of Big Data as a full-fledged field can be traced to some of the concepts implemented by these two systems – concepts such as data-intensive computing, infrastructure independence, virtualization and policy-based data management. The two software systems were developed by the Data Intensive Computing Environments group (DICE), which was started in 1994 at the San Diego Supercomputer Center (SDSC). The DICE group pursued the goal of implementing software systems that would enable collaborative research through large-scale sharing of multi-disciplinary data files. In the following we track the history of this development and the application of data grids in support of digital libraries and archives.

II. Driving Requirements

The selection of the initial software development goal was based on observations of research requirements in computational plasma physics, observations of technology management requirements within the San Diego Supercomputer Center, results from a prior collaboration on an Alternative Architecture study for the Earth Observing System [1], and research in high-performance networking within the CASA Gigabit Network project [2]. For example, in computational plasma physics, the analysis of the stability of toroidal plasma configurations was being done at institutions on the East and West coasts of the United States in the 1980s. A collaboration environment was needed to enable researchers to compare stability analyses and independently verify results. This required the ability to share input files, as well as output results, across institutional boundaries. A common name space was needed for referencing files, and descriptive metadata was needed to define a collection context.

Within the San Diego Supercomputer Center, which started in 1986, technology was replaced every three years to track and take advantage of the emergence of cheaper and higher performance systems. In particular, by 1994, the third version

This work was supported in part by the NSF OCI-1032732 grant, "SDCI Data Improvement: Improvement and Sustainability of iRODS Data Grid Software for Multi-Disciplinary Community Driven Application," (2010-2013) and in part by the NSF Cooperative Agreement OCI-094084, "DataNet Federation Consortium", (2011-2015).

Arcot Rajasekar is with the University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3360 USA (e-mail: sekar@renci.org).

Michael Wan was with the University of California, San Diego, La Jolla, CA, 92093 USA (e-mail: mwan@renci.org).

Reagan W. Moore is with the University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3360 USA (e-mail: rwmoore@renci.org).

Hao Xu is with the University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3360 USA (e-mail: xuh@cs.unc.edu).

of an archival storage system had been implemented, using a third generation of storage tape technology. A mechanism was needed to simplify migration of the archived data between old and new systems. The properties of the collections housed within the center needed to be maintained independently of the choice of storage technology.

The Earth Observing System Alternative Architecture analysis proposed that data products should be organized as a collection, and that relational database technology should be used to manage the system state information. Data replication was proposed between two centers, with data streaming to support processing of the contents. A collection-based approach to data management was expected to handle a much larger number of files than supported by the available file systems.

In the CASA Gigabit Network, theoretical predictions were made of the maximal achievable performance of a distributed, heterogeneous computational environment. The concept of superlinear speedup through the federation of heterogeneous computing resources was analyzed, and a practical demonstration was made that showed a speedup of a factor of 3.3 across two supercomputers. This indicated that management of heterogeneous resources was important for optimizing performance across distributed systems. The corresponding concept in archival storage was the use of a disk to hold small files while larger files were written to tape. This optimized access to small files while enabling the storage of massive amounts of data.

The combination of these prior research efforts pointed to the need for researchers to be able to provide a context for interpreting shared data, while managing technology evolution. These requirements for a distributed data management system were the seeds for the development of the first data grid software - the Storage Resource Broker data grid system. The same basic requirements, listed below, were also used to implement digital libraries and archives:

- **Management of data from multiple institutions as a shareable collection through virtualization mechanisms.** This was implemented by managing global name spaces for files, collections, users, and storage systems independently of the physical storage systems where the objects were stored, and independently of the administrative domains at each institution. Authentication and authorization on the global user name space were implemented as third-party services. The global name space for files mapped logically named files onto physical locations within distributed storage systems.
- **Organization of data files as a collection independently of the physical characteristics of the data file.** That is, a collection provides a virtual “grouping” of files that might be stored on distributed resources of various types, created and owned by multiple users and groups but having some common properties that warrant bundling them into the same virtual group. Not all objects in the collection need to be files, but can also be dynamic relational queries, sensor streams or self-aggregated/described objects such as tar files or HDF files.
- **Association of descriptive metadata, provenance metadata, and representation metadata with objects** in a collection to provide a context for interpreting the data and to capture domain-centric and systems-centric structured information.
- **Management of system state information in a relational database.** System metadata were associated with files, collections, users, and storage systems. This enabled rapid queries on a much richer set of attributes than normally provided by file systems. The abstraction of a common set of attributes masked the differences between the types of resources being used in the physical layer and provided a uniform system information management layer.
- **Management of the properties of the collection, independently of the properties of the storage system** in which the files were stored. This was a key goal based on the virtualization of the data collection instead of the virtualization of the storage systems.
- **Implementation of a single sign-on authentication system.** The files that were shared were owned by the data grid. Users authenticated to the data grid, and in turn, the data grid authenticated itself to the remote storage system. The files were stored under an account that represented the data grid. This meant that the data grid had to both authenticate users, and authorize actions on resources and data independently of the physical storage system. Access controls were managed by the data grid independently of the administrative domain – again providing a common service across the distributed environment.
- **An architecture based on a peer-to-peer server environment.** Users could connect to any server and the data grid would redirect the request to the correct location for the desired file operation. This meant that users could request a file without knowing where the file was located, without knowing the local name of the file (physical path name), without having an account on the remote storage system, and without knowing the network access protocol required by the storage system. The data grid managed the mapping from the logical file name to the physical path name, managed information about the file location, translated the request by the user client to the protocol required by the remote storage location, and initiated operations on behalf of the user.
- **Fault-tolerant semantics.** The intent was to build a system that tolerated failures. If a storage resource was off-line, a storage request would be redirected to alternate locations that could provide the space. This was implemented through the concept of storage resource groups. Writing to a resource group succeeded when a file was written to at least one member of the group. Thus some of the storage systems could be off-line, or down for maintenance, and the success of the operation could still be ensured. Another type of fault tolerance was achieved through replication. Since the data grid provided a mapping from the logical name to the physical address location, it was easy to extend this mapping to multiple physical addresses – hence providing management of synchronized copies of a data object distributed across multiple

resources. If access to one copy was unavailable, the system automatically provided access to its replica.

These capabilities were used to implement a data management systems that demonstrated scalability (through the dynamic addition of storage resources), access controls (through the single sign-on environment), discovery (through queries on descriptive metadata), and fault tolerance (through the use of storage resource groups).

III. Storage Resource Broker

The development of the Storage Resource Broker was funded initially by DARPA through the "Massive Data Analysis Systems" project [3] in 1995. The effort to build software to manage distributed data was viewed as a sufficiently risky objective to warrant DARPA funding. When the approach was presented at a meeting with the tape storage vendor Storage Tek, the response was that they were used to leading edge projects, but the DICE group was halfway down the cliff. The initial development integrated multiple types of technology:

- **Use of relational database technology** to manage the system state information. As part of the EOSDIS alternative architecture study (1994), a centralized architecture was proposed in which all data were managed by a relational database. The SRB data grid was designed to store system state information in a relational database, while maintaining links to files on distributed storage systems. At that time, holding and accessing hierarchical path information in relational systems was considered to be a performance bottleneck. We chose to do this in order to achieve scalability, since the file systems at that time dealt with less than 2 million files. Instances of the SRB data grid were implemented that managed over 100 million files.
- **Virtualization of data collections** versus virtualization of storage. The SRB focused on managing the properties of the data collection, instead of managing the properties of the storage systems. This made it possible to implement operations directed at data manipulation in addition to data storage. Vendors were beginning to implement storage virtualization in 1995 but considered data/collection virtualization to be too risky.
- **Support for heterogeneous storage systems.** In order to manage interactions with multiple types of storage system protocols, the SRB software was designed to map from a standard protocol that was based on extensions to POSIX I/O, to the protocol used by specific types of storage systems such as the IBM High Performance Storage System, the UniTree storage system, the Network File System, and the Cray File System etc. The protocol conversion was implemented as a modular and extensible software driver. The data grid tracked all operations performed through the middleware, and updated persistent state variables consistently within a central metadata catalog.
- **Extended support for data manipulation operations.** The SRB data grid implemented operations for replication, versioning, synchronization, auditing, aggregation in containers, staging of files, archiving of files, checksum creation, metadata extraction, and metadata loading. Since the additional operations were initiated through both Unix utilities and web browsers, a key property of the data grid was the decoupling of access mechanisms from the data management middleware.
- **Support for multiple types of client interfaces.** A second layer of virtualization was needed to manage mapping from the protocol used by client software, to the standard I/O protocol supported within the data grid. For the SRB, the clients that were supported included web browsers, Unix shell commands, Java load library, C++ I/O library, and Fortran I/O library. The protocol used by the client did not have to match the access protocol required by the storage system. In effect, the SRB implemented brokering technology between clients and storage.
- **Support for multiple authentication environments.** Since the data grid accessed resources across multiple administrative domains, it needed to deal with the different types of authentication that were supported by the collaborating institutions. To perform authentication for users to access files, multiple types of authentication systems were supported, including Unix passwords, Kerberos, and Grid Security Infrastructure through the Generic Security Service API. For each type of authentication environment, the associated information was stored in the metadata catalog as attributes on the user account name. The authentication mechanism used to authenticate a person to the data grid did not have to be the same as the authentication mechanism used to authenticate data grid access to a remote storage system. Hence, the system also worked as an authentication broker.
- **Schema indirection.** Each user community had different definitions for the descriptive metadata that they associated with files and collections. Schema indirection was used to store a triplet consisting of the attribute name, the attribute value, and an attribute unit or comment. This allowed each community to use the data grid as generic infrastructure and implement their domain specific descriptive metadata. Association of name spaces to form an entity set (e. g. Dublin Core, FITS metadata, DICOM metadata, etc.) was also possible.
- **Extensible generic infrastructure.** Since multiple types of applications built upon the SRB data grid, new features were implemented through appropriate forms of virtualization. This ensured that the system would remain compatible with prior versions, and that extensions to the software could build upon multiple versions of storage technology. The highly extensible architecture ensured long-term sustainability of the software through continued application to additional science and engineering domains.

The SRB can be viewed as an interoperability mechanism that enabled use of multiple types of storage technology, multiple types of authentication systems, and multiple types of access clients. The interoperability enabled by the SRB software is shown in Figure 1. The SRB data grid was implemented as multiple software servers that may reside on different computers or may be co-located on a single computer. Each software server ran as a user-level application on the computer. The servers communicated over a network using a protocol written specifically for the Storage Resource Broker.

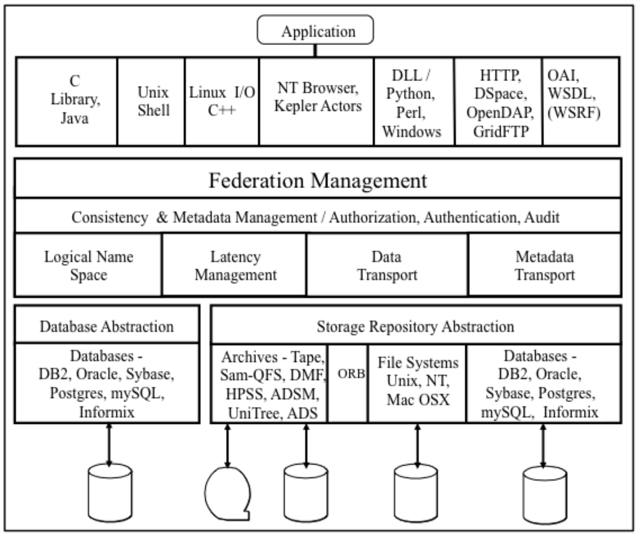

Fig. 1 Storage Resource Broker Data Grid Components

External clients accessed the data grid over a network. Each access was authenticated, and each operation was authorized by the data grid. One of the servers managed interactions with a metadata catalog, which in turn composed the SQL needed to access a relational database that stored the system attributes. The SRB had drivers for interacting with multiple types of storage systems (tape archives, file systems, objects in databases, object ring buffers) and multiple databases (DB2, Oracle, Sybase, Postgres, mySQL, and Informix). Any of the listed clients (C library, Java, Unix shell command, C++ library, web browser, Kepler workflow actor, Python load library, Perl load library, DSpace digital library, GridFTP transport tool) could discover, retrieve, or load files within the distributed environment through mapping of their API to the SRB communication protocol.

By virtualizing the data collection, it became possible to think of a digital library as the mechanism for managing the flow of technology through a permanent collection. The properties of the collection remained invariant as new technologies were selected for storing data, authenticating users, and managing access.

The development of the SRB was funded by 22 projects that represented collaborations with groups sharing data, groups managing large-scale distributed data, groups organizing digital libraries, and groups building preservation environments. The very wide range of applications ensured that generic infrastructure was developed, with appropriate virtualization mechanisms used to support the domain features of each application. See the Appendix for a list of the versions of the SRB software that were developed over a ten year period.

IV. Data Management Concepts

Within each funded collaboration project, data management concepts were developed to represent how generic infrastructure could be used to support all types of data management applications, including digital libraries and archives. The concepts are useful in that they help define standard semantics for discussing data management. In many cases, the DICE group had to invent terms, or extend the meaning of terms in order to describe what was being done. Eventually, most terms gained broader acceptance within the academic world. Each example of a concept is illustrated within the context of the collaboration project that supported the development of the associated generic infrastructure. We describe the various concepts and their timeline during the SRB development.

Logical File Name and Logical Collection (1996): In the SRB data grid, we needed a term that differentiated the name space used to organize distributed data from the names used within the physical file systems. We used the term "logical file name" to denote the identifier for a file as managed by the data grid. The "logical file name" could be organized into "logical collections", making it possible to associate files that were stored on different storage systems within the same logical collection.

Data Grid (1998): A data grid is the software infrastructure that organizes distributed data into a shareable collection. A paper describing the Storage Resource Broker data grid was presented at the CASCON conference in 1998 [4]. This paper subsequently won an award as one of the top fourteen CASCON First Decade High Impact Papers. A variant of this term was used by NASA for the Information Power Grid.

Middleware definition (1998): At an NSF middleware workshop, the question of "What is middleware?" was discussed [5]. The answer based on the SRB data grid was: "Middleware is a software system that manages distributed state information."

This definition was extended to include support for services over a network that linked the distributed environment. However, the relationship of middleware to network infrastructure was not codified in the workshop. Data grid middleware manages distributed state information about file location and file membership in collections. Networks also manage distributed state information within their routing tables about links to other routers. The resolution of this dichotomy was recently achieved within the iRODS data grid software, with the integration of policy-based data management with policy-based network routing. See the concept Software Defined Networks in Section 6.

Persistent Archive (2000): In the Transcontinental Persistent Archive Prototype, a project funded by the National Archives and Records Administration, the DICE group needed a term to describe the preservation of an archive [6]. Note that the word archive (from the computer science discipline) is used to denote the infrastructure that is used to preserve records. In the preservation community, the word "archives" is used to denote the records that are being preserved. A "persistent archive" provides a way to archive a record collection independently of the preservation environment, and then retrieve the archives for instantiation of the archive on new technology, overcoming technology obsolescence.

Preservation through Interoperability Mechanisms (2000): There is an equivalence between access to heterogeneous resources across space and access to heterogeneous resources over time. At the point in time when records are migrated to new technology, both the old technology and new technology are present. Thus data grid middleware can provide the interoperability mechanisms that enable access to both the old and the new technology [7]. The preservation infrastructure needs to provide the virtualization mechanisms that abstract preservation properties from the current choice of storage technology. In a sense, application of interoperability across spatial resources was taken to the next level by providing interoperability across time. The SRB provided a convenient mechanism for performing the temporal jumps in a seamless manner. What resulted is an "organic system" that enabled migration of data objects across time overcoming technology obsolescence through codification of infrastructure independence.

Persistent Objects (2003): Preservation communities had previously considered two basic approaches for long term preservation: 1) Emulation, in which the supporting software infrastructure was emulated to ensure that the record could be parsed using the original application; 2) Transformative migration, in which the format of the record was transformed to the format that could be parsed by modern display applications. Persistent objects is a third approach, in which the record is preserved in an unaltered form, while the preservation environment virtualizes I/O operations, enabling access to the record by modern access protocols. This viewpoint considers that the purpose of the preservation environment is to provide an interface between an original record and the ever-changing data management technology.

Consider Figure 2. Data grid technology implements persistent objects [8] by mapping from the actions requested by the display application to the protocol of the storage system where the record is located. In the iRODS data grid, this concept was extended to include the ability to write a policy in a rule language, ensuring independence from the original operating system that was used to support the policy. In both cases, the original record was not changed. Instead the preservation environment was modified to support interactions with the new technologies.

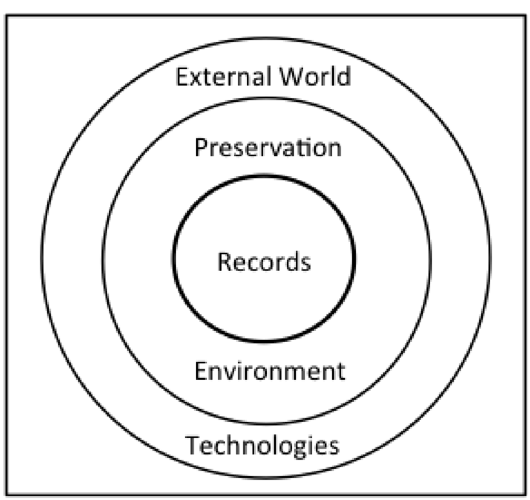

Fig. 2 Managing Technology Evolution – Persistent Objects

Policy-based Data Management (2006): One of the applications of the Storage Resource Broker was in the United Kingdom eScience Data Grid. The SRB ensured consistency by encoding within the software middleware explicit management constraints. The constraints were applied by each of the distributed servers, ensuring that the properties of the system were appropriately maintained. However, within the UK data grid, incommensurate management constraints were needed. An archive collection was desired in which no changes to records was allowed, not even by the data grid administrator. Also, a publication collection was desired in which the data grid administrator could replace bad files. Finally, a research collection was needed in which a researcher could replace files at will. Three different management policies were needed within the same data grid.

In the iRODS policy-based data management system, we identified each location in the software middleware where consistency constraints were imposed, and replaced the control software with a policy-enforcement point. On execution of the policy-enforcement point, the system would retrieve the appropriate rule from a rule base, and then execute the associated procedure. The rule controlled the procedure using state information stored in the data grid metadata catalog. Thus the rule could retrieve the name of the collection, and then enforce the appropriate deletion policy. This enables virtualization of policy management, providing both administrators and users with a declarative way to define and control actions that happen at the data storage level. Hence, one can view iRODS as defining a new generation of servers that is completely configurable and capable of enforcing user-centric actions.

Preservation as Communication with the Future (2008): The projects sponsored by the National Archives and Records Administration focused on development of an understanding of the principals behind data preservation. The traditional preservation objectives are authenticity, integrity, chain of custody, and original arrangement. These objectives are all aspects of a higher level goal, that of enabling communication with the future. The traditional representation information defined by the Open Archival Information System model provides a context for correctly interpreting a record by a future knowledge community through creation of preservation metadata. In the future, the knowledge community will have enough information from the associated representation information to correctly interpret a record. This viewpoint needed to be augmented with a characterization of the representation information that describes the preservation environment itself. Within policy-based data management systems, the environment representation information is characterized by the policies and procedures that are used to manage the records along with the associated system state information. It is then possible for an archivist in the future to verify communication from the past, and validate that the preservation objects have been appropriately preserved [9].

If preservation is communication with the future, then policy-based systems enable verification of the validity of communication from the past. The same concept can be applied to digital libraries. In this case, the librarian makes assertions about the properties of the digital library, such as completeness and consistency. A future librarian should be able to verify that the library properties have been conserved over time.

V. Integrated Rule Oriented Data System

In 2006, the Storage Resource Broker development was deprecated, in favor of developing an Open Source version of data grid technology. At the same time, a decision was made to go beyond data and information virtualization, to also support knowledge virtualization. The basic approach was to turn policies into computer actionable rules, turn procedures into computer executable workflows, and use policy enforcement points to decide when policies should be applied.

The architecture of the policy-based data management systems was similar to the SRB, as shown in Figure 3.

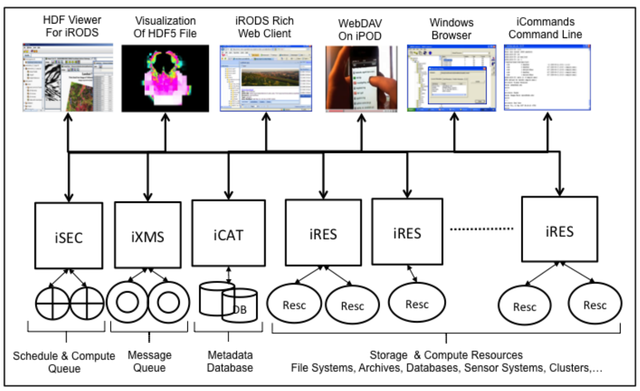

Fig. 3 Policy-based Data Management Architecture

Multiple peer-to-peer servers managed interactions with remote storage locations, and a central metadata catalog stored state information in a relational database. The integrated Rule-Oriented Data System (iRODS) also implemented servers to manage message passing, and to manage a queue of outstanding rule requests [10].

A comparison of policy-based systems with distributed data management systems shows how the concepts related to data management have been evolving. Figure 4 illustrates the central concepts behind traditional file systems, and also

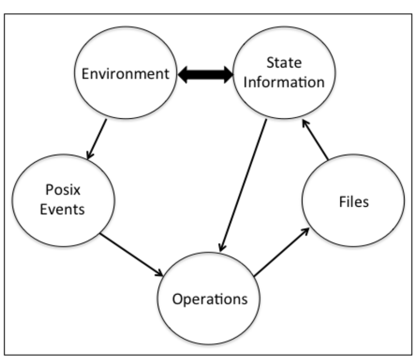

Fig. 4 File System Characterization

behind the Storage Resource Broker. External events interact with the data management system through a well defined protocol. The data management system uses state information to control the execution of operations on the stored files, and the state information is appropriately updated. The file system (i-nodes, v-nodes, etc.) environment in some sense is synonymous with the state information that is managed about the files. A key component of a file system is the consistent update of the state information after every operation that is performed upon the files. The SRB answered the challenge of self-consistent update of state information in a distributed environment, across heterogeneous storage systems, across multiple administrative domains.

In policy-based data management systems, operations are replaced by policies that control updates through procedures, and files are replaced by objects that may include workflows, active or realized objects, and databases, as well as files. Figure 5 lists the characteristics of policy-based data management, representing the evolution from traditional file-based systems to information and knowledge based systems.

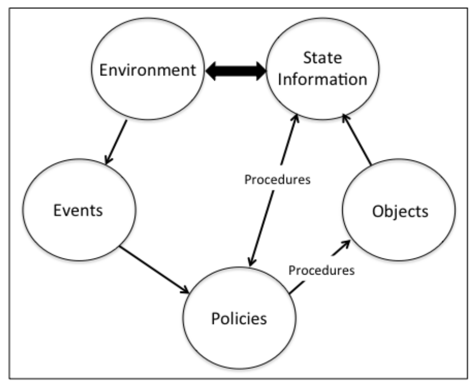

Fig. 5 Policy-based System Characterization

As before, the data management environment is synonymous with the consistent management of state information. However, in the policy-based system, the environment is governed by a set of policies that are implemented as computer actionable rules. Thus a description of the environment must include not only the state information, but also a listing of the policies and procedures that are being enforced. Similar to the evolution of concepts for the SRB, the development of iRODS also required several new concepts, which we describe along with a timeline. See the Appendix for a list of the versions of the iRODS software that have been released at the time of this document.

Computer Actionable Knowledge (2006): A major goal of data grid technology has been the extension of data management systems to also support information management and knowledge management through computer actionable forms. The Storage Resource Broker augmented data management with information management, by associating state information as metadata attributes on an appropriate name space. The types of information that were managed included system administration information, provenance information, descriptive information, and representation information.

Policy-based data management systems augment information management with knowledge management. The knowledge required to execute a protocol, or manipulate a file, or access a remote repository is encapsulated in a microservice. The microservices can be chained together to implement a workflow, or procedure. Policies control when and where each procedure can be executed. In a sense, a file (or object) is not viewed in isolation, but along with all the policies and procedures that govern its creation and usage. The application of knowledge requires the dynamic execution of procedures. The result of the execution is stored as system state information, and is assigned as metadata on objects within a name space. In essence, the reification of a knowledge procedure is turned into administrative information that is stored as metadata in a relational database. One can view the metadata as inherent properties (labels) on the objects that codify the derived knowledge obtained through application of procedures.

This approach to knowledge management through computer actionable forms can be quantified as follows:

- Data consists of bits (zeros and ones)
- Information consists of labels applied to data
- Knowledge evaluates relationships between labels
- Wisdom evaluates relationships between relationships

Within the iRODS data grid, data are managed as files in a file system, or objects in an object store. Information is managed as metadata in a relational database. Knowledge is applied as computer actionable rules through a rule engine. Wisdom (within the confines of the user-configurable iRODS system) is applied through policy enforcement points which determine when and where the knowledge procedures should be executed.

Note that the concept of relationships has been extended to include:

- Semantic or logical relationships
- Spatial or structural relationships
- Temporal or procedural relationships
- Functional or algorithmic relationships
- Systemic or epistemological relationships

Thus a procedure is the application of a functional relationship to a digital object to generate either information about the digital object, or a new digital object [11].

The differentiation between information and knowledge is complex. In order to assign a label to a digital object, a knowledge relationship between existing labels needs to be evaluated. However each existing label required the prior application of knowledge relationships. Information generation is an infinite recursion on the application of knowledge procedures. Each knowledge procedure evaluates relationships between labels that were previously generated. The recursive nature is closed by reducing the information labels to a well known set that are interpreted the same way by the entire user community. The simplest way to separate information and knowledge is to view information as the reification of knowledge. Information is a static property, while knowledge is the active evaluation of a relationship.

Our first attempt to characterize information and knowledge was expressed as a matrix, with the goal of differentiating between ingestion, management, and access services for digital objects [12]. This characterization focused on services that were used to manipulate data, information and knowledge, within the context of a data grid. Figure 6 shows the components of the characterization, with the data grid represented by the matrix that links together the individual components related to the types of service.

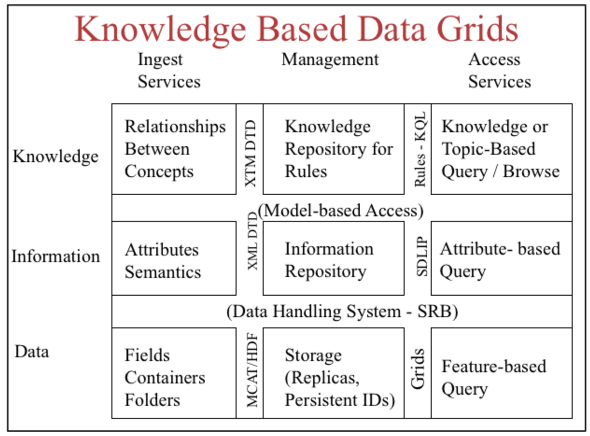

Fig. 6 Knowledge-based Grids

This characterization is realized in the iRODS policy-based data management system. The services to manipulate data are the operations supported upon digital objects. The storage systems for data are accessed through storage drivers. The services to manipulate information are the operations supported upon metadata attributes. The information repository is the metadata catalog, stored in a relational database. The knowledge relationships between concepts are implemented by chaining microservices that are controlled by computer actionable rules. The knowledge repository is implemented as a rule base. The knowledge-based grid can be viewed spatially, as a shared distributed service or temporally, as a persistent archive.

The access services remain an area of active development, and are further discussed in the feature-based indexing concept in Section 6.

Knowledge Virtualization (2010): The iRODS data grid provides virtualization of data, information, and knowledge. Figure 7 shows a simple architecture view of the

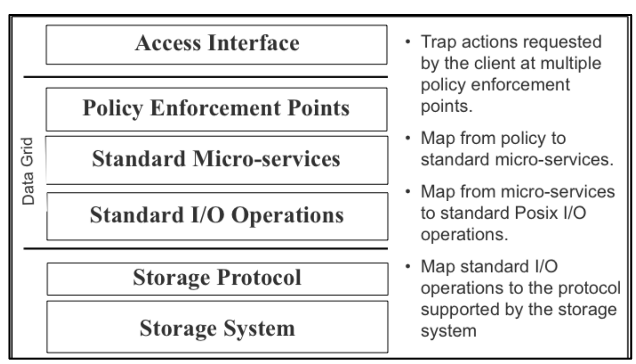

Fig. 7 iRODS data grid virtualization mechanisms

interoperability mechanisms. An access interface virtualizes access by mapping from the access protocol to the iRODS interaction protocol. Each interaction is trapped at policy enforcement points where a rule base is consulted to determine which policy to execute. The policies control the execution of procedures that are composed by chaining together basic functions, called microservices. This requires that the middleware manage exchange of structured information between the chained microservices through in-memory structures. When the microservices are executed on different servers, the information structures are serialized, moved over the network, and unpacked into in-memory structures at the remote system.

The microservices perform operations such as I/O manipulation, metadata extraction, and domain-specific operations. Each microservice invokes standard POSIX-based I/O operations. The data grid middleware then translates between the standard I/O and the protocol required by the remote storage location. Thus the microservices are independent of the operating system. The same microservices run on Windows, Unix, and Mac computers, enabling the migration of policies and procedures across operating systems. The ability to manage application of knowledge procedures, independently of the choice of storage environment, can be viewed as a form of knowledge encapsulation.

Policies as Intellectual Property (2013): A major goal of the development of policy-based data grid middleware has been the conversion of management policies into computer actionable rules that control computer executable procedures. This enabled multiple communities, shown below, to apply the technology. The users of the software span multiple science and engineering disciplines, and include national data grids, national libraries, and international projects:

Archives
  Taiwan National Archive, Digital Preservation Network
Astrophysics
  Auger supernova search
Atmospheric science
  NASA Langley Atmospheric Sciences Center
Biology
  Phylogenetics at CC IN2P3
Climate
  NOAA National Climatic Data Center
Cognitive Science
  Temporal Dynamics of Learning Center
Computer Science
  GENI experimental network
Cosmic Ray
  AMS experiment on the International Space Station
Dark Matter Physics
  Edelweiss II
Data Grids
  Bestgrid, French Grid Initiative
Digital Libraries
  French National Library
Earth Science
  NASA Center for Climate Simulations
Ecology
  CEED Caveat Emptor Ecological Data
Engineering
  CIBER-U
High Energy Physics
  BaBar / Stanford Linear Accelerator
Hydrology
  Institute for the Environment, UNC-CH; Hydroshare
Institutional Repository
  Carolina Digital Repository
Genomics
  Wellcome Trust Sanger Institute
Libraries
  French National Library, Texas Digital Libraries
Medicine
  Lineberger Comprehensive Cancer Center
Neuroscience
  International Neuroinformatics Coordinating Facility
Neutrino Physics
  T2K and dChooz neutrino experiments
Oceanography
  Science Observatory Network
Optical Astronomy
  National Optical Astronomy Observatory
Particle Physics
  Indra multi-detector collaboration at IN2P3
Plant genetics
  CyVerse
Quantum Chromodynamics
  IN2P3
Radio Astronomy
  Cyber Square Kilometer Array, TREND, BAOradio
Seismology
  Southern California Earthquake Center
Social Science
  Odum Institute for Research in Social Science, TerraPop

Each community implemented different choices for semantics, policies, and procedures. A generalization of the observed usage patterns is to associate the intellectual properties of each community with the policies and procedures that they implemented. The underlying data grid middleware was generic infrastructure that provided the mechanisms needed to virtualize interactions with data, information, and knowledge. The policies and procedures encapsulated the knowledge that was needed to apply the middleware within each domain.

This means that intellectual property can be captured and applied within generic data management infrastructure to cater to the specific needs of each domain. This idea is extended in Figure 8, which describes a general approach towards quantifying intellectual property.

Each domain is characterized by:

- **Purpose** driving the formation of a data collection. The purpose represents a consensus of the persons collaborating on a data management project.
- **Properties** that will be maintained for the data collection. The properties are dependent upon the driving purpose. If the

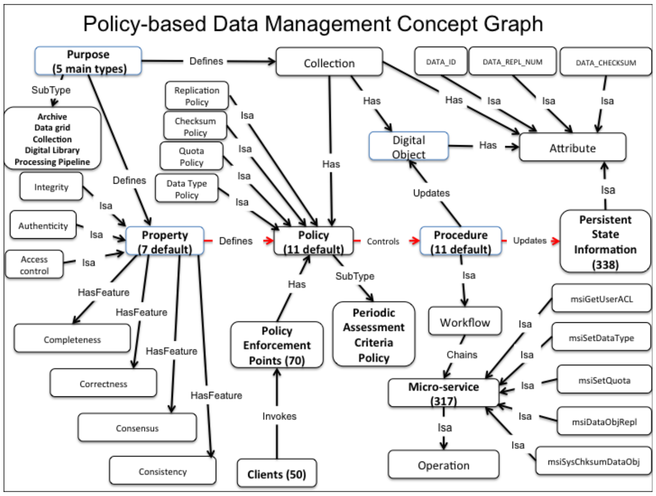

Fig. 8 Conceptualizing intellectual property as policies and procedures

intent is preservation, then properties related to authenticity, chain of custody, integrity, and original arrangement are desired. If the intent is formation of a digital library, then properties related to descriptive metadata, file arrangement, and file format may be desired. The properties comprise assertions made about the collection by the developers of the collection. Other domain centric elements (such as provenance, retention, disposition, etc.) can also be defined as part of these properties.

- **Policies** that enforce the desired properties. The policies control when and where management procedures are executed. Multiple policies may be needed for each desired property. In general, policies are needed to control generation of the desired property. Policies are also needed to validate whether the desired property has been maintained over time. Since the distributed environment is subject to multiple forms of risk (network outage, storage system maintenance, operator error, policy change), assessment criteria are needed that can be checked to verify compliance with the desired collection properties. The assessment policies are turned into computer actionable rules that are periodically executed. Example domain centric policies include enforcing authority (e. g. HIPAA policies), integrity checks, data cleansing, metadata extraction, etc..
- **Procedures** codify and apply the operations needed to generate a desired property. Examples include procedures to create a replica, extract metadata, set access controls, manage a quota, check a retention period, apply disposition, etc. Procedures are executed as computer executable workflows.
- **Persistent state information** is generated each time a procedure is run. The persistent state is stored as metadata attributes on one of the name spaces managed by the data grid. The state information can be queried for compliance at a point in time. To verify compliance over time, the system parses audit trails. Persistent state information in turn codify the properties of a collection.

A viable policy-based data management system must be sufficiently sophisticated to handle a wide variety of data management applications. The iRODS data grid provides 317 microservices that can be used to compose procedures, and manages 338 persistent state information attributes. In practice, each domain implements a small number of policies. Out of the box, the iRODS data grid source provides 14 default policies for enforcing data sharing properties. Communities typically add another 5 policies on the average to control desired features. However, the range of policies that are required to support a fully customized data grid may be very large.

Each policy and procedure set encapsulates the domain knowledge needed to manage a specific domain application.

Federation through Interoperability Mechanisms (2014): Within the DataNet Federation Consortium [14], the iRODS data grid is being used to create national data cyberinfrastructure through the federation of existing data repositories. In the process, interoperability mechanisms have been implemented that enable three basic federation approaches:

1. Tightly coupled federations. The name spaces used to identify users and files are shared between two data grids. A data grid can store and retrieve files in a second data grid through middleware servers that enable application of the desired operations at the remote repository. In effect, the operations are moved to the data.

2. Loosely-coupled federations. The knowledge needed to interact with a remote data management system is encapsulated in a microservice that retrieves data from the remote repository using the protocol of the remote repository. This is a traditional approach similar to brokering, in which data are retrieved for analysis at the local computer. The data are moved to the processing engine.

3. Asynchronous federations. No direct interaction occurs between the federated data repositories. Instead an intermediary (such as a message bus queue) is used to hold requests that have been encapsulated in messages. Requests for an operation are posted to the queue. The remote system retrieves messages from the queue, does the desired operations, and posts results back to the queue.

Using these three mechanisms, the DataNet Federation Consortium has been able to support interoperability with web services, sensor networks, union catalogs, data repositories, workflow environments, databases, message buses, and systems that communicate over the internet.

The expectation is that these three federation mechanisms are sufficient to federate all existing data management applications. The DataNet Federation Consortium currently (2016) federates systems across national projects in oceanography, cognitive science, plant biology, engineering, hydrology, and social science.

Quantifying the Broadening of Impact through a Collection Life Cycle: A notable requirement for National Science Foundation funding is the demonstration that the research results will impact a broad user community. A mechanism has been needed to quantify the impact. One way to do this has been the observation that the set of policies and procedures used to manage a collection evolve over time to represent the current requirements of each broader user community. It is possible to quantify impact by tracking the policy evolution. This can be represented through a collection life cycle:

- **Project collection** – usually the team members have complete tacit knowledge about the acceptable semantics, data formats, and analysis procedures used with the project data sets. The data sets are organized in a project collection with minimal metadata. The data sharing is limited to the group, and is mostly through shared and mounted file spaces.
- **Shared collection (data grid)** – when data products are shared with other groups and institutions, the tacit knowledge must be made explicit. Policies are needed to govern the application of semantic terms, and the transformation of data to required data formats. Policies are also needed to enforce authentication, access controls and data distribution. Policies for data manipulation may also be needed.
- **Published collection (digital library)** – when the results are formally published for use by the discipline, policies are needed to enforce domain standards for semantics and data formats. Policies are also needed to generate persistent identifiers, to validate integrity, and to track provenance.
- **Processing pipeline** – when the data sets are used in an analysis service, procedures are needed that support the manipulation and transformation of the data. The provenance of derived data products needs to be captured.
- **Preserved reference collection (archive)** – when the results are archived for use by future researchers, a sufficient context is needed that enables a person in the future to interpret the data. The knowledge is typically reified in representation information. At the same time, the policies and procedures of the preservation environment also need to be preserved so a future archivist can verify that the collection was managed correctly.

The broadening of user impact can be quantified through the evolution of the policies and procedures that are used to manage the information context associated with a data collection. A digital library thus represents the publication stage of a collection life cycle. The policies associated with a digital library map from the assertions made by the group that formed the collection, to the expectations for discovery and access of the members of the discipline. This mapping is resolved in terms of required metadata, required data formats, required processing procedures, and required user interfaces.

Policy Sets (2015): Data management applications are governed by policies and procedures. In collaboration with the Practical Policy working group of the Research Data Alliance, the DFC analyzed the policies needed to implement representative systems for data sharing (data grids), data publication (digital libraries), production data centers, data preservation (archives), management of protected data, and NSF Data Management Plans [16]. The approach was based on identifying the tasks that needed to be done for each type of data management application, and then developing representative policies for automating each task. Three types of policies were created: policies to set system parameters to control execution of the task; policies to manage the task execution, and policies to verify tasks were executed correctly.

Across the six categories of data management, a total of 97 tasks was identified. The tasks required the use of 119 operations and 50 persistent state information attributes. The operations included workflow operators to control processing, collection manipulation, file manipulation, user account management, system parameter settings, storage resource access, and metadata manipulation. The persistent state information attributes included information about collection properties, file properties, metadata, quotas, resource properties, system parameters, user properties, and data grid properties.

The expectation is that generic policy sets can be defined for each type of data management application that can be modified for use by a specific institution. This will greatly simplify the automation and control of data collections, and enable the development of systems that automate auditing and validation mechanisms.

VI. Future Data Management Infrastructure

The current generation of data grid middleware is still evolving. New opportunities to apply policies to control the data management environment are emerging. We consider three specific extensions, the inclusion of policies within storage controllers, the integration of policy-based data management with policy-based networks, and the extension of a knowledge grid into a wisdom grid.

Feature-Based Indexing: A major challenge in constructing a collection is the assignment of appropriate descriptive metadata. This is a laborious task, which potentially is non-scalable. A major question is whether the act of description can be turned into the application of a knowledge procedure, that is automatically applied within the storage system. Normally descriptive metadata are used to provide a context for the contents of a file. An alternative approach is to use descriptive metadata to define features present within a file. If the desired features can be extracted by a knowledge procedure, then the generation of descriptive metadata can be automated.

This approach is being explored in collaboration with storage vendors. The Data Direct Networks storage controllers now support virtual machine environments that can be used to run the iRODS data grid. When a file is written to the storage system, the data grid can apply feature extraction procedures automatically, and index the stored data by the features present within each record. Hence, one can construct a domain-centric data-grid appliance that can perform automated data management including automated data description.

Software Defined Networks (2013): Policy-based systems are also appearing within networks that are based on the OpenFlow router. Network routing decisions can be controlled by policies that are used to manage path-selection within the router. A demonstration of the use of policy-based data grids to control policy-based routing was given at the Supercomputing '13 conference [15]. The iRODS data grid managed information about the location of files, their access controls, and the availability of replicas. Within the iRODS data grid, a parallel data transfer was set up, with subsets of the file sent in parallel over multiple network paths. The iRODS data grid communicated with the OpenFlow router to select a disjoint network path for each of the parallel data transfer channels.

The idea here is that a traditional data grid views the network as a black box (and vice versa, the network is opaque with respect to the applications at the end-points of the communication pipeline). If the data grid is able to export some of its policies to be implemented by the network (through the OpenFlow router) and also is able to get feedback from the routers about network topology, congestion and statistics, the two can work together to mutual advantage and improve performance. Having this exchange of information can be used in multiple ways to improve data grid operations.

One way to exchange information is through the integration of control policies between data grids and networks. Since both systems are managing distributed state information, it is reasonable to think about formally moving data grid middleware into network routers. It will then be possible to access data by name (or metadata attribute) instead of an IP address, enforce access controls within the network, cache data within the network, and debug data transfers by single-stepping through the data grid procedures (currently supported in iRODS).

The approach would rely upon the data grid to provide a context for the files through their organization in collections. A file would be referenced by its membership in a collection, with the data grid controlling the access (authentication and authorization). The data grid would negotiate with the network for selection of the replica to use as the starting point, and the network path to use for data delivery. In the long term, data grid middleware should disappear as separate infrastructure, and be subsumed within the network. The upshot of this would be collection-oriented addressing of objects instead of name-oriented or ip-oriented addressing for data ingestion, movement and access.

The integration of software-defined networks with data grids enables the virtualization of data flows. The properties of a data flow can be managed, independently of the type of network. This would include naming of data flows, re-application of a data flow, access controls on data flows, and sharing of data flows. A content delivery system could be defined as a data flow which is re-executed periodically.

Wisdom management: Current virtualization mechanisms focus on data, information, and knowledge. Future data management systems will also need to support virtualization of wisdom. If we can think of wisdom as the evaluation of relationships between relationships, then we can build a computer actionable form of wisdom. Within the iRODS data grid, wisdom is captured as hard-coded policy-enforcement points that control when and where knowledge procedures are executed. To make application of wisdom a dynamic process, the system will need to implement mechanisms that enable wisdom-based decisions to be selected as systemic processes that apply to all interactions. This will require processing information about each access session, information about the collections, and information about the user community to infer which set of knowledge procedures should be applied.

Within the iRODS Consortium, a pluggable version of the iRODS data grid has been developed. New microservices can be added dynamically to implement new operations. When the microservice is plugged into the framework, two policy enforcement points are dynamically created to control pre-process rules and post-process rules. With this approach, the operations performed by the data grid can be separated from the middleware framework. A pre-process rule can be created which generates an event message that is posted for processing within an external indexing system. A post-process rule can be created that tracks all changes to the system state information and posts update messages. This means that all changes to the state information within the data grid can be tracked and associated with the corresponding client action. The compliance of the system to the desired policies can be verified.

In effect, the application of wisdom procedures is reified as event information. Procedural and temporal relationships can be evaluated across all of the event information, enabling the application of wisdom procedures to the events that occur within the data management system. A digital library should be able to track all events, apply reasoning across the events to detect usage patterns, and adjust policies to optimize user interactions.

The iRODS software was hardened and modularized, and is now maintained and distributed, by the iRODS Consortium, beginning with the 4.0.0 Release. Information about the consortium, ongoing development, and future planned releases is available at http://irods.org.

Appendix

iRODS Releases

iRODS 4.1.9 July 28, 2016
iRODS 4.1.8 February 22, 2016
iRODS 4.1.7 November 20, 2015
iRODS 4.1.6 October 1, 2015
iRODS 4.1.5 September 2, 2015
iRODS 4.1.4 August 5, 2015
iRODS 4.1.3 June 18, 2015
iRODS 4.1.2 June 5, 2015
iRODS 4.1.1 June 2, 2015
iRODS 4.1.0 May 29, 2015
JSON-based configuration, Dynamic PEPs
iRODS 4.0.3 August 20, 2014
iRODS 4.0.2 June 17, 2014
iRODS 4.0.1 June 5, 2014
iRODS 4.0.0 March 28, 2014
Binary Packages, Pluggable architecture
iRODS 3.3.1 February 24, 2014
SHA2 hash, Rule looping, WSO extensions
iRODS 3.3 July 17, 2013
NetCDF support, HDFS, PAM authentication

iRODS 3.2 October 3, 2012
WSO objects, direct access resources
iRODS 3.1 March 16, 2012
Tickets, locks, group-admin updates
iRODS 3.0 September 30, 2011
New rule language, soft links
iRODS 2.5 February 24, 2011
Database resources, Fortran I/O library
iRODS 2.4 July 23, 2010
Bulk upload, monitoring,
iRODS 2.3 March 12, 2010
Extensible iCAT, quotas, group-admin
iRODS 2.2 October 1, 2009
HPSS driver, S3 driver, compound resource
iRODS 2.1 July 10, 2009
mySQL driver, Kerberos, policy enforcement
iRODS 2.0 December 1, 2008
Federation, master/slave catalog, bundling
iRODS 1.1 June 27, 2008
GSI, mounted structured files, HDF5, Jargon
iRODS 1.0 January 23, 2008
Oracle driver, FUSE interface, rule language
iRODS 0.9 June 1, 2007
Replication, metadata, trash, integrity checking
iRODS 0.5 December 20, 2006
Policy enforcement points, rule engine

SRB Releases

SRB 3.5 Dec 3, 2007
Bind variables, bulk replication, transfer restart
SRB 3.4 Oct 31, 2005
Master/slave MCAT, HDF5 integration
SRB 3.3 Feb 18, 2005
ACL inheritance, bulk move, GT3 GSI
SRB 3.2 July 2 2004
Client initiated connections, Database access
SRB 3.1 April 19, 2004
Synchronization, trash can, checksums
SRB 3.0 Oct 1, 2003
Federation
SRB 2.0 Feb 18, 2003
Parallel I/O, bulk load, metadata access control
SRB 1.1.8 Dec 15, 2000
Encrypted passwords, large file size
SRB 1.1.7 May 2000
GSI authentication
SRB 1.1.6 Nov 1999
Stream support, Oracle support
SRB 1.1.4 May 1999
Containers
SRB 1.1.3 Feb 1999
Recursive replication
SRB 1.1.2 Dec 1998
Monitoring daemon
SRB 1.1 Mar 1998
Query support
SRB 1.0 Jan 1998
Unix commands

Acknowledgment

The original members of the DICE group included Reagan Moore, who led the group; Michael Wan, who was the chief architect and designed the communication and management protocols for the peer-to-peer server architecture; Arcot Rajasekar, who proposed and implemented key innovations related to virtualization, management of state information, and policy-based data management; Wayne Schroeder, who implemented the security environment, unix-style utilities, and the testing environment; Lucas Gilbert, who implemented a Java I/O library; Sheau-Yen Chen, who was the grid administrator for the group through multiple evolutions of the grid; Bing Zhu, who ported the system to Windows; and Chaitan Baru, Richard Marciano, Ilkay Altintas, Bertram Ludaescher, and Amarnath Gupta who applied the technology. The DICE group maintained a core set of developers for twenty years, while adding expertise including Mike Conway, who developed advanced Java library interfaces; Hao Xu, who optimized and extended the iRODS distributed rule engine, and Antoine de Torcy, who developed iRODS microservices for application domains.

The SRB and iRODS technologies were developed and applied across more than 30 funded projects, and multiple funding agencies. These included:

NSF HydroShare	7/1/12-6/30/17
NSF EarthCube Layered Architecture	4/1/12-3/31/13
NSF DFC Supplement Extensible Hardware	9/1/11-8/31/15
NSF DFC Supplement Interoperability	9/1/11-8/31/15
NSF DataNet Federation Consortium	9/01/11-8/31/16
NSF SDCI Data Improvement	10/1/10 – 9/30/13
NOAA National Climatic Data Center	10/1/09 – 9/1/10
NSF Temporal Dynamics of Learning	1/1/10 – 12/31/10
NARA Transcontinental Persistent Archive	9/15/9-9/30/10
NSF Temporal Dynamics of Learning	3/1/9-12/31/9
NSF Transcontinental Persistent Archive	9/15/8-8/31/13
NSF Petascale Cyberfacility Seismic Com.	4/1/8-3/30/10
NSF Data Grids for Community Driven Apps	10/1/7-9/30/10
DOD Joint Virtual Network Centric Warfare	11/1/6-10/30/7
NSF Petascale Cyberfacility for Seismic Data	10/1/6-9/30/9
LLNL Scientific Data Management	3/1/5 – 12/31/08
NARA Persistent Archives	10/1/4-6/30/08
NSF Constraint-based Knowledge	10/1/4-9/30/6
LC NDIIPP California Digital Library	2/1/4-1/31/7
NASA Information Power Grid	10/1/3-9/30/4
NARA Persistent Archive	6/1/2-5/31/5
NSF National Science Digital Library	10/1/2-9/30/6
DOE Particle Physics Data Grid	8/15/1-8/14/4
NSF SCEC Community Modeling	10/1/1-9/30/6
DOE Terascale Visualization	9/1/98-8/31/02
NSF Grid Physics Network	7/1/00-6/30/05
NARA Persistent Archive	9/99-8/00
NSF Digital Library Initiative UCSB	9/1/99-8/31/04
NSF Digital Library Initiative Stanford	9/1/99-8/31/04
NASA Information Power Grid	10/1/98-9/30/99
NSF NPACI data management	10/1/97-9/30/99
DOE ASCI	10/1/97-9/30/99
DARPA Massive Data Analysis Systems	9/1/95-8/31/96

References

[1] F. Davis, W. Farrell, J. Gray, R. Mechoso, R. Moore, S. Sides, and M. Stonebraker, "EOSDIS Alternative Architecture", Hughes Applied Information Systems, Tech. Rep. Contract # ECS-00012, September 6, 1994.

[2] R. Moore, "Distributing Applications Across Wide Area Networks", General Atomics, San Diego, CA, Tech. Rep. GA-A20074, May 1990.

[3] R. Moore, C. Baru, R. Frost, R. Marciano, A. Rajasekar and M. Wan, "MDAS - A Massive Data Analysis System", in *Proc. of Interface97 Symposium*, 1997.

[4] C. Baru, R. Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker," in *Proc. CASCON'98 Conference*, Nov.30-Dec.3, 1998, Toronto, Canada, p. 5.

[5] B. Aiken, J. Strassner, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moore, B. Teigelbaum, "Network Policy and Services: A Report of a Workshop on Middleware", The Internet Society, 2000.

[6] R. Moore, C. Baru, A. Rajasekar, B. Ludaescher, R. Marciano, M. Wan, W. Schroeder, and A. Gupta, "Collection-based Persistent Digital Archives – Part 2", D-Lib Magazine, April 2000.

[7] R. Moore, C., Baru, A. Rajasekar, B. Ludaescher, R. Marciano, M. Wan, W. Schroeder, and A. Gupta, "Collection-Based Persistent Digital Archives – Part 1", D-Lib Magazine, March, 2000.

[8] R. Moore, "The San Diego Project: Persistent Objects," Archivi & Computer, Automazione E Beni Culturali, l'Archivio Storico Comunale di San Miniato, Pisa, Italy, February, 2003.

[9] R. Moore, "Towards a Theory of Digital Preservation", IJDL Volume 3, Issue 1, August 2008, pp. 63-75.

[10] A. Rajasekar, M. Wan, R. Moore, and W. Schroeder, "A Prototype Rule-based Distributed Data Management System", Presented at HPDC workshop on "Next Generation Distributed Data Management", May 2006, Paris, France. Available: https://wiki.irods.org/pubs/DICE_RODS-paper.pdf

[11] R. Moore, "Automating Data Curation Processes", in Proc. of *NSF Curating for Quality Workshop*, September 11, 2012, Arlington, Virginia, pp. 67-71. Available: http://datacuration.web.unc.edu

[12] R. Moore, "Knowledge-based Grids," Proc. *18th IEEE Symposium on Mass Storage Systems and Ninth Goddard Conference on Mass Storage Systems and Technologies*, San Diego, April 2001.

[13] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, S.-Y. Chen, L. Gilbert, C.-Y. Hou, C. Lee, R. Marciano, P. Tooby, A. de Torcy, and B. Zhu, "iRODS Primer: Integrated Rule-Oriented Data System", Morgan & Claypool, 2010.

[14] R. Moore and A. Rajasekar, "Reproducible Research within the DataNet Federation Consortium", Proc. *International Environmental Modeling and Software Society 7th International Congress on Environmental Modeling and Software*, San Diego, California, June 2014, http://www.iemss.org/society/index.php/iemss-2014-proceedings.

[15] S. Huang, H. Xu, and Y. Xin, "A Framework for Integration of Rule-oriented Data Management Policies with Network Policies", Proc. *3rd GENI Research and Educational Experiment Workshop* [GREE 2014], Atlanta, GA, March 2014.

[16] R. Moore, A. Rajasekar, and H. Xu, "DataNet Federation Consortium Policy Toolkits", *iPRES conference*, November 2015.

Arcot Rajasekar was born in India. He received the B.S. degree in Electronics and Communications from the University of Madras, the M.S. degree in Computer Science from the Indian Institute of Technology in Madras, and the Ph.D. degree in Computer Science from the University of Maryland.

He was an Assistant Professor at the University of Tennessee and joined the DICE group in 1996. He is a lead originator behind the concepts in the SRB and iRODS data grids. He is a professor in the School of Library and Information Science at the University of North Carolina at Chapel Hill.

Michael Y.-K. Wan was born in Hong Kong in 1950. He received the B.S. degree from Illinois State University in 1972 and the M.S. degree from Georgia Institute of Technology in Nuclear Engineering.

He was a staff member of the San Diego Supercomputer Center from 1986 to 1994, working on operating systems and archival storage systems. From 1994 to 2008, he was a member of the DICE group at SDSC. He led the Data Intensive Cyber Environment group of the Institute of Neural Science at the University of California, San Diego, from 2008 to 2014.

Wan is the chief software architect of the integrated Rule-Oriented Data System and the Storage Resource Broker.

Reagan W. Moore was born in Ventura, California in 1946. He received the B.S. degree in Physics from the California Institute of Technology in 1967, and the Ph.D. degree in Plasma Physics from the University of California at San Diego in 1978.

From 1974-1986 he was a Staff Scientist at General Atomics. From 1986-1994 he was in charge of production services at the San Diego Supercomputer Center, and managed the Data Intensive Computing Environments group from 1994 to 2008. From 2008-2015 he led the Data Intensive Cyber Environments Center at the University of North Carolina at Chapel Hill, where he is a Professor in the School of Information and Library Science.

Professor Moore is co-author of three U. S. patents on "Elongated Toroid Fusion Device" (1988), "Transparent management of data objects in containers" (2004), and "Persistent Archives" (2005).

Hao Xu was born in China. He received his Ph.D. degree from the University of North Carolina at Chapel Hill in 2014, and joined the DICE Center as staff.

Xu is the architect of the pluggable rule engine used in the iRODS data grid.