Policy-based Data Grids

Reagan W. Moore[1], Arcot Rajasekar[1], Michael Wan[3], Wayne Schroeder[2], Antoine de Torcy[1], Sheau-Yen Chen[2], Mike Conway[1], Hao Xu[1]
[1]University of North Carolina at Chapel Hill,
[2]University of California, San Diego
[3]San Diego, California

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 (2004-2014) [http://irods.org]. Applications included national data grids, national digital libraries, 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 foundational concepts are built upon evolutionary concepts in virtualization and the abstractions needed to manage data, information, and knowledge.

I. Introduction:

The implementation of two successful data grid software systems – the Storage Resource Broker (SRB) and the integrated Rule Oriented Data System (iRODS), represents an example of a software development life cycle. User requirements from the academic science community drove the implementation of data grid technology, and the evolution data grids from data management to information and knowledge management. These 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) to pursue the goal of implementing generic data management systems that would enable collaborative research through large-scale sharing of multi-disciplinary data files.

Five foundational concepts form the basis for generic data management infrastructure:

1. Development of computer actionable forms of data, information, and knowledge
2. Development of an architecture that can maintain consistent state information across a distributed environment
3. Development of virtualization and interoperability mechanisms needed to interact with exiting technologies
4. Development of a policy-based system to differentiate control policies from generic operations
5. Generic implementation across traditional data management applications

We review each foundational concept and discuss how the concepts influenced the design of data grid technology.

II. Computer actionable forms of data, information, and knowledge:
Modern data management systems must manage not only data, but contextual information about each digital entity. In addition, the systems must capture related knowledge about formation of the digital entity. To do this, we define the following computer actionable representations:

- Data are traditionally represented as digital entities stored in files, or rows in a database, or objects in an object store, or web pages on a web site. A generalization of the concept of a digital entity is the association of an operation with a "logical" name. The execution of the operation generates the desired digital entity.
    - A file can be registered as a digital entity. Clicking on the logical name causes the system to retrieve the file from a remote data grid server.
    - A soft link can be registered as a digital entity. Clicking on the logical name causes the system to invoke the appropriate procedure for retrieving the referenced file from a remote repository, through use of the required protocol.
    - A query on a database can be registered as a digital entity. Clicking on the logical file name causes the system to apply the query against the database, and retrieve a file containing the result of the query.
    - A workflow can be registered as a digital entity. Clicking on the logical name causes the workflow to be executed. The system can save the provenance information for each execution of the workflow (input parameters, input files) and also save the output. This enables reproducible data-driven research.
- Information corresponds to the assignment of a *name* or *label* to a digital entity. The choice of the appropriate *name* to assign is derived by evaluating knowledge relationships. These relationships may be semantic or logical, spatial or structural, temporal or procedural, functional or algorithmic, and systemic or epistemological. If the relationships are satisfied, the information can be assigned as a metadata attribute associated with the digital entity, and stored in a relational database. We needed a generalization of the concept, with each *name* being given associated attributes, such as attribute values, attribute units, comments, creation date, creator name, etc. Schema indirection was used to manage the addition of new attribute names and associated attribute properties. This makes it possible to provide descriptive metadata, provenance metadata, administrative metadata, and representation metadata to any digital object in the collection, and also to user names, storage resource names, rule names, and function names.
- Knowledge corresponds to the evaluation of relationships between names. This implies that knowledge must be captured as an active process or

procedure. The execution of a procedure corresponds to the application of a specific set of knowledge. The outcome of the procedure can be saved as information assigned to a digital entity. Within the iRODS data grid, knowledge is managed as computer executable workflows controlled by computer actionable policies. Note that knowledge is inherently recursive. The names that are used in a relationship are themselves governed by subsidiary relationships between subsidiary names. The recursion is stopped through reification of knowledge as information.

- Wisdom corresponds to the evaluation of relationships between relationships. Typical types of computer actionable wisdom correspond to deciding when (temporal relationship) and where (structural relationship) to evaluate knowledge procedures. Within the iRODS data grid, wisdom is encoded as policy enforcement points. When a client action causes a policy enforcement point to be executed, associated knowledge procedures are automatically applied.

Using these abstractions, it then becomes possible to have a data management system manage data, information, and knowledge.

III. Persistent state architecture

The original application supported by the DICE group was shared collections assembled across distributed storage systems. The goal was to support access to shared data without knowing where a digital entity was stored, without having an account at the remote storage location, without knowing the name of the file at the remote repository, and without knowing the access protocol used by the remote repository. The data grid was required to track the administrative information needed to handle all interaction properties. In particular, the data grid was implemented as middleware, software systems that manage distributed state information.

In distributed environments, a major challenge is the consistent management of state information across all operations that may be performed. The result of each operation must be tracked and the state information appropriately updated. To ensure consistency in a distributed environment, a central logical catalog interface was designed that managed interactions with a database for storing the state information. Even though all updates were sent through the logical catalog interface, the underlying database technology could be distributed, sharded, replicated, and parallelized. In effect, the requirement for a central logical catalog interface meant that all the metadata were logically deposited in a single basket, and the system could monitor interactions with the single basket very closely to ensure consistency across all distributed operations. In practice, while writes are to a single logical catalog interface, reads could be done independently on slave catalogs. Thus it was easy to distribute reads of state information.

In distributed environments, a peer-to-peer server architecture is needed to ensure users can interact with a local system for improved access performance. This meant that a user could connect to any of the peer systems, and the data grid would

forward the request to the logical central catalog, authenticate the user and authorize the operation, identify the appropriate peer for applying the operation, forward the request to the remote server, execute the operation, and return the response. A consequence of this approach is that the data grid "owns" the shared data. The data grid applies operations on behalf of the user, after authentication and authorization are satisfied.

IV. Virtualization and interoperability mechanisms

To enable access by modern clients to legacy storage repositories (that have an inflexible, legacy access protocol), two levels of virtualization were needed. The actions requested by a client were mapped to a high level protocol. The high level protocol actions (such as replication, copy, checksumming, metadata extraction) were implemented in basic functions (micro-services) that executed a low-level protocol based on extensions to Posix I/O (open, close, read, write, seek, stat, …). The low level protocol was then mapped to the protocol required by a remote storage system. These two levels of virtualization ensured that new clients could be added without having to change how storage systems were accessed. Also, new storage systems could be incorporated without having to change how the clients were accessed.

A second benefit was that the basic actions performed within the data grid were operating system agnostic. The same micro-service could run on any of the operating systems to which the data grid had been ported. This also meant that the basic functions were persistent, they could continue to be applied in the future on new operating systems. The system could be sustained across arbitrary generations of technology, and was "infrastructure independent".

A third benefit was that it was possible to define three basic interoperability mechanisms that enabled integration of new technology into the data grid. These were:

1. The ability to interact with a remote repository using the protocol of the remote repository. The interaction was encoded in a micro-service that could be invoked as part of a procedure. This meant that data could be organized into a shareable collection without any changes to the remote repositories.
2. The ability to apply the data grid operations at the remote repository. While this required the installation of software middleware at the remote location, it then became possible to do data manipulation before moving a file.
3. The ability to control the distributed environment through specified management policies and procedures. The policies decide when and where operations are applied, enabling the enforcement of data grid policies across the distributed environment, independently of the remote repositories.

V. Policy-based Data Management

Policy-based systems abstract control mechanisms from the underlying operations. This makes it possible to build generic infrastructure that is controlled by policies

that are unique to a specific community or data management application. At each place within the data grid middleware where a constraint or consistency check should be applied, a policy-enforcement-point (PEP) is inserted. When a PEP is encountered in the code execution, the data grid accesses a distributed rule engine to determine which policy should be applied, and then executes the procedure associated with the policy. The policy explicit specifies the constraints that must be satisfied before the procedure can be run.

An interesting observation is that most policies are relatively static. They correspond to assertions that the community building the shared collection wants to have enforced. Thus policies tend to be updated only after a community consensus is reached. For performance, each server in the data grid runs a local rule base that is read by a local rule engine. Updates to the rule bases are managed from through central logical catalog interface.

By adding appropriate policy-enforcement-points, policies can be enforced across any desired action by any client. Since the PEPs are part of the middleware, policies are enforced independently of the type of client. Adding a new client will not change the enforcement of the management policies.

VI. Applications of policy-based systems

Once policies have been abstracted from the generic infrastructure, it becomes possible to change the data management application by changing the policies that are stored in the rule base. Thus the iRODS data grid has been applied to support preservation (policies for authenticity, integrity, chain of custody, and preservation of the arrangement of records in a record series); digital libraries (policies for arranging data in collections, access controls, descriptive metadata selection); data grids for sharing data (policies for data distribution, integrity, access controls, provenance); data processing pipelines (policies for controlling workflows, workflow provenance, data caching); and data distribution systems (policies for distribution, caching, retention, disposition).

VII. Future development

Policy-based systems are now being implemented in data management middleware, storage systems (policies to control replication and distribution), networks (policies to manage data flows), and clients (policies to modify data before transmission). Each system is applying management policies to control the entities within a particular set of infrastructure. This is a significant change from original operating systems, which focused on controlling a resource (interactions with disk, CPUs, memory). Now data grids are focused on managing the entities that use disk, CPUs, and memory. This corresponds to virtualization of data collections (instead of storage systems), virtualization of workflows (instead of compute resources), and virtualization of network flows (instead of network routers). Policy-based systems enable the management of the properties of collections, workflows, and data flows.