iRODS Micro-Services

Reagan Moore

{moore, sekar, mwan, schroeder, bzh, ptooby, antione, sheau}@dicereseearch.org
{chienyi, marciano, michael_conway}@email.unc.edu
Implications

- iRODS policies are enforced at the remote storage location
- Equivalent of a distributed operating systems is needed
  - State information
  - In-memory data structures
  - Message system
  - Rule queuing
  - Scheduling
  - Remote execution
iRODS - Distributed Operating System
Simplification

- Compose well-defined procedures
  - Control execution of procedures through computer actionable rules
- Remote procedures constructed by chaining micro-services together
  - Micro-services are functions encoded in C
  - Strongly "typed", explicit knowledge of the information structures used by each micro-service
  - Explicit names for state information
List of Micro-services (~185)

- irule -F listMS.ir

List_Available_MS
{
    msiListEnabledMS(*KVPairs)
    writeKeyValPairs(stdout, *KVPairs, ": ")
}

INPUT *A=null
OUTPUT ruleExecOut
Micro-Service Examples

- msiDataObjRepl
- msiDataObjCopy
- msiGetObjType
- msiAssociateKeyValuePairsToObj
- msiExtractTemplateMDFromBuf
- msiCollCreate
- msiNoTrashCan
- delayExec
- remoteExec
- forEachExec
- msiSleep
- writeLine
Variables

- Variables, used to describe input and output parameters
  - Labeled with an “*”
- Session variables, used to define attributes related to the session
  - List of available session variables in iRODS Primer
  - Labeled with a “$”
- Persistent state variables, used to define attributes that are permanently stored in iCAT metadata catalog
  - List of available persistent state variables in iRODS Primer
Session Variables

- Availability depends upon the action that is being performed
- Interactive rule execution provides a limited set of session variables
  - userNameClient
  - rodsZoneClient
- If invoke an action related to file manipulation, get session variables for
  - objPath
  - replNum
  - dataSize
  - chksum
Persistent State Variables

- Can be listed using the icommand
  - iquest attrs
- Examples include
  - DATA_NAME
  - DATA_SIZE
  - DATA_CHECKSUM
  - DATA_PATH
  - DATA_REPL_NUM
  - DATA_RESC_NAME
  - DATA_VERSION
Implication - Structured Information

- Each micro-service ingests and outputs structured information
  - Explicit in-memory data structures defined for exchanging structured information
  - Need to check micro-service is being given the correct data structure type
- Doxygen lists the structures used for each micro-service
Data Checksum Micro-service

- `msiDataObjChksum` (msParam_t * inpParam1,
  msParam_t * msKeyValStr,
  msParam_t * outParam,
  ruleExecInfo_t * rei)

  - **[in]** inpParam1 - A DataObjInp_MS_T or a STR_MS_T which would be taken as dataObj path.
  - **[in]** msKeyValStr - Optional - a STR_MS_T. This is the special msKeyValStr format of `keyWd1=value1++keyWd2=value2++keyWd3=value3...` If the keyWd is not specified (without the '=' char), the value is assumed to be the target resource ("destRescName") for backward compatibility. Valid keyWds are:
    - "ChksumAll" - checksum all replicas. This keyWd has no value. But the '=' character is still needed.
    - "verifyChksum" - verify the chksum value.
    - "forceChksum" - checksum data-objects even if a checksum already exists in iCAT.
      This keyWd has no value.
    - "replNum" - the replica number to checksum. This keyWd has no value.
  - **[out]** outParam - a STR_MS_T containing the chksum value.
  - **[in,out]** rei - The RuleExecInfo structure that is automatically handled by the rule engine.
Infrastructure Independence

- Micro-services manipulate structures in memory
  - iRODS framework maps from requested I/O operations to Posix-style I/O
  - iRODS drivers map the Posix-style I/O to the specific storage protocol
- Implication
  - Same micro-service runs on Windows, Unix, Linux, Mac operating system
  - Procedures can be executed across any of the linked operating systems
Migration of Micro-services

Map from actions requested by the access method to a standard set of Micro-services.

Map the standard Micro-services to standard operations.

Map the operations to protocol supported by the operating system.
Wednesday afternoon

- Session on writing a new micro-service
- Developing a book that will list for each micro-service:
  - Input parameters / structures
  - Output parameters / structures
  - Persistent state information that is set
  - Operations performed upon files