Performance of iRODS

Reagan W. Moore

University of North Carolina at Chapel Hill

rwmoore@renci.org

Topics

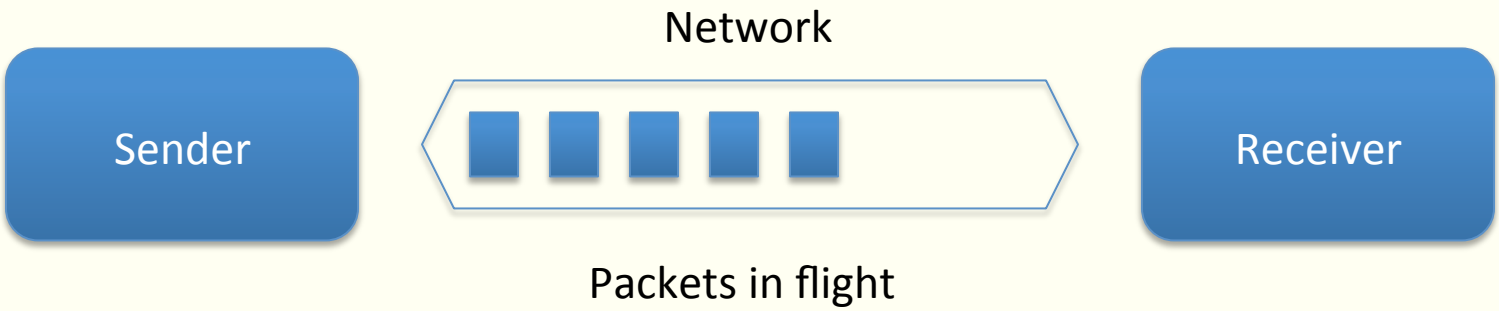
- Data transfer performance
- Metadata catalog performance
- Rule engine performance
- High performance systems

Data Transfer

- **Independent assessments published at** https://www.irods.org/index.php/Performance
- **Assessments are dependent upon:**
  - Storage systems
  - TCP/IP tuning
  - Network bandwidth headroom
  - Choice of protocol
  - System load

Observations

- **Performance is driven by whether the network pipe can be filled**
  - Require enough messages in flight that there is no additional delay caused by waiting for an acknowledgement packet



Optimizations

- **For small files, encapsulate data in request to send**
  - File size less than 32 MegaBytes
- **For large files, use parallel I/O streams**
  - N Numthreads option on iPut
  - Set window size for number of messages in flight
- **For reliable networks, can use RBUDP**
  - Reliable Blast UDP limited to a maximum of 1000 tries on network disconnect
- **Caveats**
  - iRODS can keep track of progress, and restart transmission after network disconnect
  - Users in the iPlant Collaborative have moved 2.5 TeraByte files over the network

Transfer Rates

- https://www.irods.org/index.php/Lyon-KEK
  - Describe transfer from KEK, Japan to Lyon, France
  - Achieved transfer rates between 10-45 MB/sec for moving a 1 GB file with window size of 4 MB and 16 I/O streams
- **Typically observe best performance when**
  - Use 2 I/O streams
  - Send multiple files in parallel (multiple iput commands)

iCAT Performance

- https://www.irods.org/index.php/File:rt1.png
- **Observe**
  - Time to ingest a 100-byte file varied from 10 milliseconds to 45 milliseconds as number of files in collection increased to 10 million
  - Tests done on pre-iRODS release 1.0
- **Optimizations:**
  - Index the database periodically
  - Distribute load across multiple databases

Rule Engine Performance

- **Tested time to execute rules on my Mac**
  - Ran Ubuntu in a Virtual Box emulator on 2.53 GHz Mac Book Pro Intel Core i5
  - iRODS version 3.0
- **Computed time to:**
  - Execute a simple micro-service within the rule engine
  - Make a database query
  - Loop over logic needed for a production quality rule
- **Compared results to disk spin latency time**
  - 11 milliseconds

Production Integrity Rule

- *Verify all input parameters for consistency.*
- *Query the iRODS metadata catalog to retrieve status information*
- Verify the integrity of each file in a collection
- Update all replicas to the most recent version.
- *Minimize the load on production services through a deadline scheduler*
- *Differentiate between the logical name for a file and the physical replica locations.*
- Identify all missing replicas and *document their lack*.
- Create new replicas to replace missing replicas.
- *Implement load leveling* to distribute the new replicas across the storage systems
- *Create a log file that records all repair operations performed upon the collection.*
- T*rack progress of the policy execution.*
- *Initialize the rule for the first execution.*
- *Enable restart of the process from the last set of checked files in case of a system halt.*
- *Manipulate files in batches of 256 files at a time to handle arbitrarily large collections.*
- *Minimize the number of sleep periods used by the deadline scheduler.*
- *Include the checking of new files that have been added during the execution of the policy*
- *Write out statistics* about the effective execution rate, and the number of files checked.

Find Where Replicas are Stored

```
# get all replica numbers for this file
      msiMakeGenQuery("DATA_REPL_NUM,DATA_CHECKSUM,DATA_RESC_NAME", "COLL_NAME =
'*Colln' and DATA_NAME = '*Name'", *GenQInp4);
      msiExecGenQuery(*GenQInp4, *GenQOut4);
      *Numr = 0;
      *Ulist = *Ulist0;
      foreach(*GenQOut4) {
       *Numr = *Numr + 1;
       msiGetValByKey(*GenQOut4, "DATA_REPL_NUM", *Repln);
       msiGetValByKey(*GenQOut4, "DATA_CHECKSUM", *Chk);
       msiGetValByKey(*GenQOut4, "DATA_RESC_NAME", *Rescn);
       msiDataObjChksum("*Colln/*Name", "replNum=*Repln++++forceChksum=", *Chkf);
       if(int(*Chk) == 0) {
         *Chk = *Chkf;
       }
     }
```

Workflow Operations Used

- **Arithmetic (+, -, *, /)**
- **Boolean tests (==, !=, &&, ||, >, <, >=)**
- **Conditional statements**
  - if / then / else
- **Control**
  - break / fail
- **Loops**
  - for / foreach / while
- **List manipulation**
  - initialization / list addition (cons) / extracting an element from a list (elem) / updating an element in a list (setelem)
- **Variable manipulation**
  - initialization / type conversion (int, double, str)

Micro-services Used

- **Metadata catalog manipulation**
  - msiGetValByKey — get metadata from structure
  - msiExecStrCondQuery — execute string conditional query
  - msiString2KeyValPair — convert string to key-value pair
  - msiAssociateKeyValuePairsToObj — add metadata
  - msiMakeGenQuery — create a query
  - msiExecGenQuery — execute a query
  - msiCloseGenQuery — release query buffers
  - msiGetContInxFromGenQueryOut — check for more rows
  - msiRemoveKeyValuePairsFromObj — remove metadata
  - msiGetMoreRows — get more rows from query

Micro-services Used

- **Data and directory manipulation**
  - msiIsColl — check whether name is a collection
  - msiCollCreate — create a collection
  - msiDataObjCreate — create a file
  - msiDataObjRepl — replicate a file
  - msiDataObjChksum — checksum a file
  - msiDataObjUnlink — delete a file

- **System functions**
  - msiGetSystemTime — get the system time
  - writeLine — write a line to a file or standard out
  - msiSleep — sleep

Times:

- **Initiate rule engine and loop over a counter**
  - 35 micro-seconds
- **Query metadata catalog**
  - 714 micro-seconds
- **Query metadata catalog, return results in batches of 256 rows, extract metadata from each row**
  - 155 micro-seconds

Production Rule Timing

- **Validate integrity of each file**
  - Calculate checksum and compare to value in iCAT
- **Identify missing replicas**
  - Replace missing files, load leveling across storage systems
- **Write log file, track progress, enable restart**
- **Required query to iCAT for each file**
  - Time per file was 6.3 milliseconds with no checksum
  - Time per file was 18.8 milliseconds with checksum

High Performance Systems

- **Performance depends upon the server used to support the metadata catalog**
  - PGPool to distribute load across multiple servers
- **DDN SFA12KE**
  - Includes virtual machine environment within the storage controller
  - Run iRODS and iCAT within the storage controller
  - Target is > 2000 files ingested per second.
  - Depends on using distributed database running in multiple virtual machines