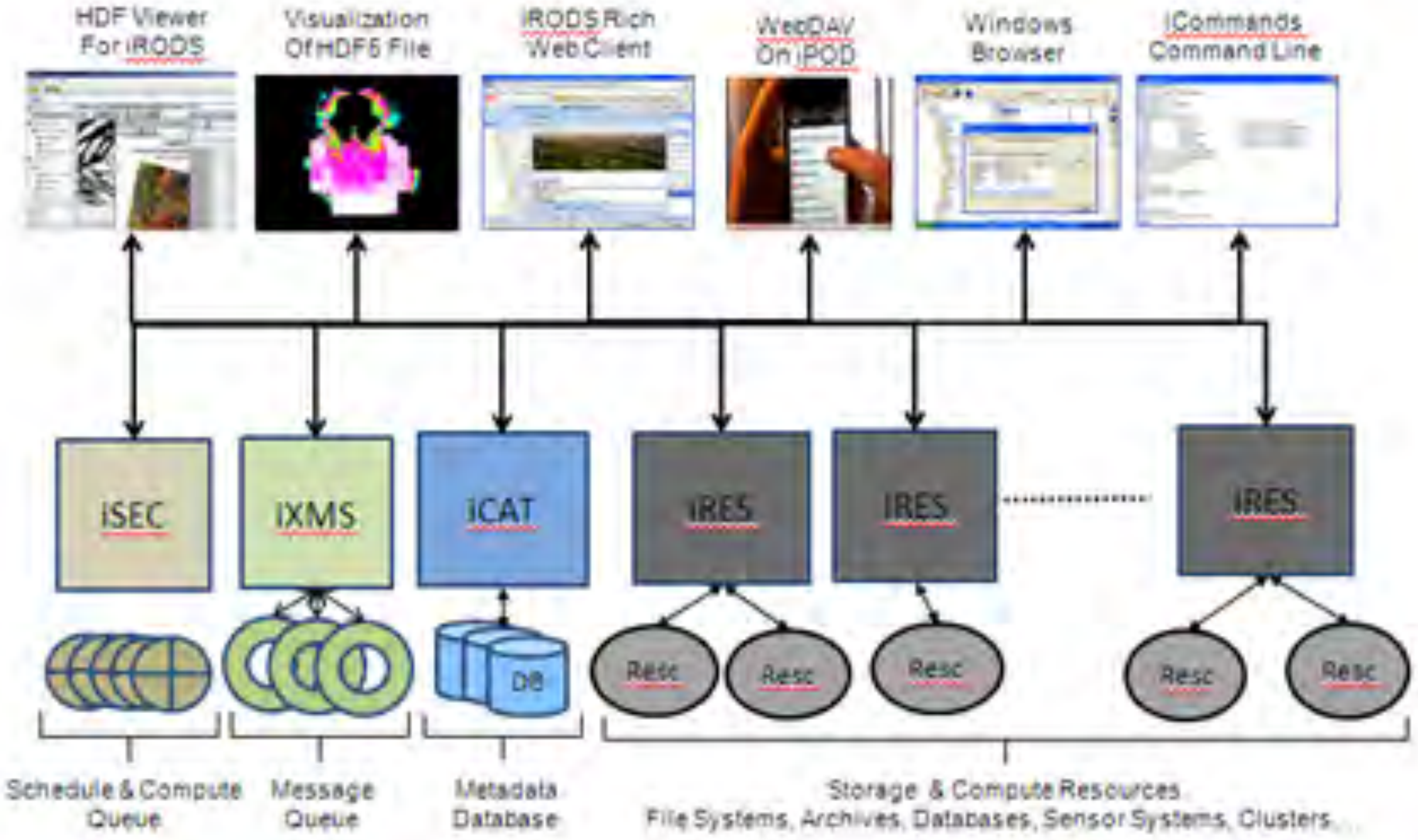
iXMS: A Message Server in iRODS

What is iXMS

- It is a Messaging Server
  - A Post Office that receives and distributes messages
  - Provides for Asynchronous Communication
    - Loosely coupled exchange of information between co-operating processes
  - Messages are queued and in FIFO-fashion
  - Key part of Message-Oriented Middleware Architecture
  - Somewhat similar to JMS, MSMQ, WebSphereMB, …

iRODS Distributed Data Management



D·I·C·E iRODS THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL UCSD

What does it do?

- Provides message exchange service
  - Create a message stream (queue)
    - Returns a “ticket”
  - Send message to a stream [put]
  - Receive message from a stream [get]
  - Destroy a stream
  - Clear a stream [restart]
- Provides some rudimentary stream control
  - How many “reads” are allowed per message
  - List of receivers who can read a message
  - How long to keep a message (expiry)
  - Get a message based on some “condition”

How to Use: API

- conn = rcConnectXmsg (&myRodsEnv, &errMsg);
  - Connect to message server. Similar connecting to iRODS server
- status = clientLogin(conn);
  - Get authenticated
- status = rcSendXmsg (conn, sendXmsgInp);
  - Send a message
- status = rcRcvXmsg (conn, &rcvXmsgInp, &rcvXmsgOut);
  - Receive a message
- status = rcDisconnect(conn);
  - Disconnect from server

How to use: iCommand

SEND

- ixmsg s [-t ticketNum] [-n startingMessageNumber]
  [-r numOfReceivers] [-H header] [-M message]

RECEIVE

- ixmsg r [-n NumberOfMessages] [-t tickefStreamtNum]
  [-s startingSequenceNumber] [-c conditionString]

CREATE A STREAM

- ixmsg t

DROP/DESTROY A STREAM

- ixmsg d -t ticketNum

CLEAR A STREAM

- ixmsg c -t ticketNum

=== Add to .irodsEnv ===

```
xmsgHost 'srbbrick14.sdsc.edu'
xmsgPort 1235
```

How to Use: micro-services

LOW-LEVEL MICRO-SERVICES

- msiXmsgCreateStream
- msiCreateXmsgInp
- msiSendXmsg
- msiRcvXmsg
- msiXmsgServerDisConnect

HIGH-LEVEL MICRO-SERVICES

- writeXMsg
- readXMsg

FUNCTIONS USEFUL FOR INCLUDING IN MICRO-SERVICES

- _writeXMsg
- _readXMsg

DEFAULT STREAMS

- Some streams (id) are started when Xmessage Server starts
  - 1 can be used to pipe stdout
  - 2 can be used to pipe stderr
  - 3 used for sending micro-service audit trail
  - 4 used for “distributed debugging” of rule execution (under development)
  - 5 unused now.

Rule Engine Audit

```
rods@tempZone:srbbrick14:11950#22::idbug:ApplyRule: acChkHostAccessControl
rods@tempZone:srbbrick14:11950#23::idbug:  GotRule: acChkHostAccessControl
rods@tempZone:srbbrick14:11950#24::idbug:ApplyRule: acSetPublicUserPolicy
rods@tempZone:srbbrick14:11950#25::idbug:  GotRule: acSetPublicUserPolicy
rods@tempZone:srbbrick14:11950#26::idbug:ApplyRule: acAclPolicy
rods@tempZone:srbbrick14:11950#27::idbug:  GotRule: acAclPolicy
rods@tempZone:srbbrick14:11950#28::idbug:ApplyRule: acPostProcForDataObjWrite(*WriteBuf)
rods@tempZone:srbbrick14:11950#29::idbug:  GotRule: acPostProcForDataObjWrite(*WriteBuf)
rods@tempZone:srbbrick14:11950#30::idbug:ApplyRule: acSetRescSchemeForCreate
rods@tempZone:srbbrick14:11950#31::idbug:  GotRule: acSetRescSchemeForCreate
rods@tempZone:srbbrick14:11950#32::idbug:    ExecMicroSrvc: msiSetDefaultResc(demoResc,null)  DATAIN:/tempZone/home/rods/tg/iinit.c
rods@tempZone:srbbrick14:11950#33::idbug:ApplyRule: acRescQuotaPolicy
rods@tempZone:srbbrick14:11950#34::idbug:  GotRule: acRescQuotaPolicy
rods@tempZone:srbbrick14:11950#35::idbug:    ExecMicroSrvc: msiSetRescQuotaPolicy(off)
rods@tempZone:srbbrick14:11950#36::idbug:ApplyRule: acSetVaultPathPolicy
rods@tempZone:srbbrick14:11950#37::idbug:  GotRule: acSetVaultPathPolicy
rods@tempZone:srbbrick14:11950#38::idbug:    ExecMicroSrvc: msiSetGraftPathScheme(no,1)  DATA:/tempZone/home/rods/tg/iinit.c  RESC:demoResc
rods@tempZone:srbbrick14:11950#39::idbug:ApplyRule: acPreProcForModifyDataObjMeta
rods@tempZone:srbbrick14:11950#40::idbug:  GotRule: acPreProcForModifyDataObjMeta
rods@tempZone:srbbrick14:11950#41::idbug:ApplyRule: acPostProcForModifyDataObjMeta
rods@tempZone:srbbrick14:11950#42::idbug:  GotRule: acPostProcForModifyDataObjMeta
rods@tempZone:srbbrick14:11950#43::idbug:ApplyRule: acPostProcForCreate
rods@tempZone:srbbrick14:11950#44::idbug:  GotRule: acPostProcForCreate
rods@tempZone:srbbrick14:11950#45::idbug:ApplyRule: acPostProcForPut
rods@tempZone:srbbrick14:11950#46::idbug:  GotRule: acPostProcForPut
rods@tempZone:srbbrick14:11950#47::idbug:    ExecMicroSrvc: msiSysReplDataObj(nvoReplResc,null)  DATA:/tempZone/home/rods/tg/iinit.c
                                                                                                 DATAIN:/tempZone/home/rods/tg/iinit.c  RESC:demoResc
rods@tempZone:srbbrick14:11950#48::idbug:ApplyRule: acSetMultiReplPerResc
rods@tempZone:srbbrick14:11950#49::idbug:  GotRule: acSetMultiReplPerResc
rods@tempZone:srbbrick14:11950#50::idbug:ApplyRule: acSetRescSchemeForCreate
rods@tempZone:srbbrick14:11950#51::idbug:  GotRule: acSetRescSchemeForCreate
rods@tempZone:srbbrick14:11950#52::idbug:    ExecMicroSrvc: msiSetDefaultResc(demoResc,null)  DATAIN:/tempZone/home/rods/tg/iinit.c
rods@tempZone:srbbrick14:11950#53::idbug:ApplyRule: acPreprocForDataObjOpen
rods@tempZone:srbbrick14:11950#54::idbug:  GotRule: acPreprocForDataObjOpen
rods@tempZone:srbbrick14:11950#55::idbug:ApplyRule: acSetVaultPathPolicy
rods@tempZone:srbbrick14:11950#56::idbug:  GotRule: acSetVaultPathPolicy
rods@tempZone:srbbrick14:11950#57::idbug:    ExecMicroSrvc: msiSetGraftPathScheme(no,1)  DATA:/tempZone/home/rods/tg/iinit.c  RESC:nvoReplResc
```

How fast is it?

Future: Xmessage System

- More Applications
  - Parallel Micro-service execution
  - Distributed Debugging
  - Delayed Execution
  - Support usage by the iRODS Community
- Improvements
  - Persistence
  - Non-authenticated streams
  - Push Methods
  - More conditions
  - Binary Messages
  - Standards Compliant Interfaces (AMQP, JMS, …)
  - Message Analysis Service (server-side processing)

Questions?