JG|U

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

Development of the iRODS-RADOS resource plugin

Matthias Grawinkel (grawinkel@uni-mainz.de)
Zentrum für Datenverarbeitung
Johannes-Gutenberg University Mainz

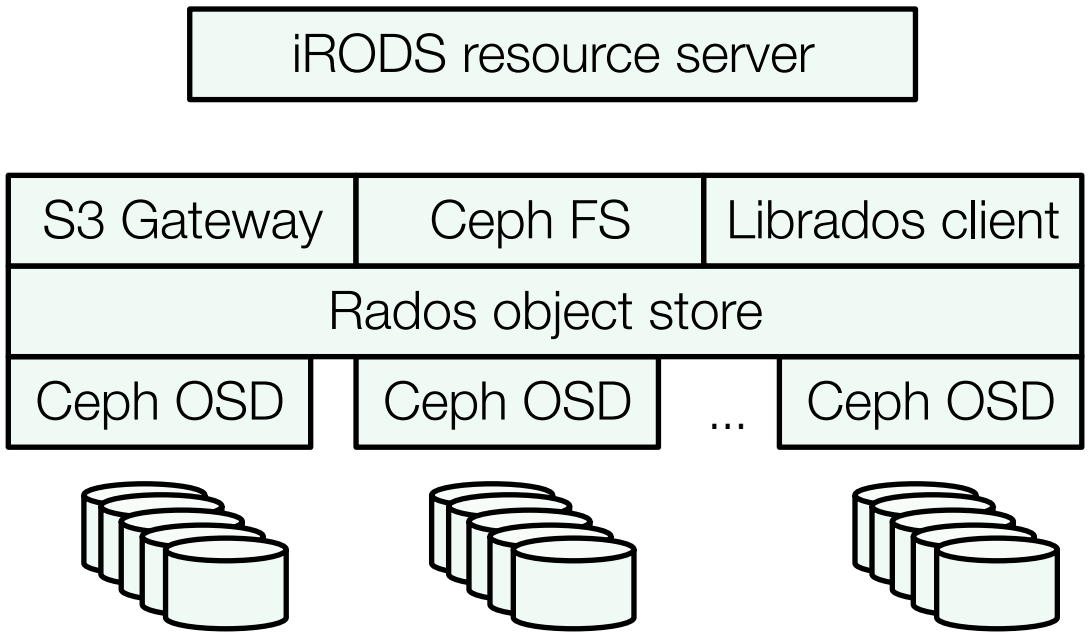
2014-06-18

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Motivation

- Building an iRODS based archival system for research data management at Johannes-Gutenberg University of Mainz.
- Integration and use of existing storage solutions
- Evaluation of ceph based storage cluster
  - No best practices for iRODS + ceph



iRODS + ceph?

- ceph is great!
  - Flexible, fast, robust, scaling storage system framework
- iRODS S3 plugin + Rados S3 Gateway
  - No stable S3 plugin for e-irods till iRODS 4.0.0
    - Place a staging file system in front of ceph?!
    - Compound Resource cache + archive
- Ceph FS will provide POSIX file system
  - Maybe a good solution
  - Adds file system overhead to rados store
  - Not stable yet
- Direct access to rados object store?
  - Let‘s try that!

Concept

- Minimize layers between iRODS resource server and rados
- Based on iRODS file system plugin
  - POSIX like fs calls
  - create(), open(), read(), write(), close(), rename(), unlink()
  - Data is organized in local filesystem

- librados – Client to rados cluster
  - Key value store
    - object_id -> blob
    - + user attributes
  - read(), write(), append(), remove()
  - Data is organized in pools
    - Client capabilities r/w per pool
    - Quota (max objects / max bytes)
    - #Replicas, distribution policies, ...

File Names & Pathes

- Every copy of a file in iRODS has two pathes
  - Logical: /zone/home/user/file
  - Physical: /path/to/storage/zone/home/user/file
- Mapping file system tree to flat object namespace (key->value)
  - Use /the/full/path as key to blob?
    - Long keys
    - Maintenance of moves?
      - imv /old /new
    - Rados cannot rename a key or move and object
- Use unique identifier?
  - uuid
  - Hash(content)
    - Hash is known after file is transmitted -> staging required
  - hash(logical path)
    - Rename operations...

File Names & Pathes 2

- File creation generates uuid as rados key

```
std::string oid = rand_uuid_string();
irods::file_object_ptr fop =
        boost::dynamic_pointer_cast< irods::file_object>( _ctx.fco() );
fop->physical_path(oid);
```

- What about directories?
  - iRODS manages namespace operations
  - Are opendir(), readdir(), closedir() required?
  - Can be implemented with some overhead
    - Store logical path as attribute to rados objects
    - Manage file system like directory blocks?
    - Update on create, rename, unlink operations

State in Stateless Architecture

- New plugin instance on rs for every client session
- Context for each file/stream
  - Logical / physical path
  - File descriptor

```
irods::file_object_ptr fop =
        boost::dynamic_pointer_cast< irods::file_object >( _ctx.fco() );
```

- Property map per plugin instance
  - Track file descriptor‘s offset in property map
    - Seek, read, write

```
int fd = fop->file_descriptor();
uint64_t read_ptr = 0;
_ctx.prop_map().get < uint64_t > ("OFFSET_PTR_" + fd, read_ptr);

…
_ctx.prop_map().set < uint64_t > ("OFFSET_PTR_” + fd, (read_ptr));
```

State in Stateless Architecture 2

- Ceph cluster connection + io_ctx instance required for access
  - Singleton, lazy initialization per plugin instance

```
irods::error e =
        _ctx.prop_map().get<librados::IoCtx*>("CEPH_IOCTX", io_ctx);
if (e.code() == KEY_NOT_FOUND) {
    connect_rados_cluster();
    …
}
```

- Synchronous reads and writes

```
librados::bufferlist write_buf;
write_buf.append((char*)_buf, _len);
int status = io_ctx->write(oid, write_buf, _len, write_ptr);
```

```
int status = io_ctx->stat(oid, &psize, &pmtime);
```

Evaluation Setup

- Plugin requires iRODS >= 4.0.3
- Client
  - 20 GB Ram Disk
    - To prepare files for upload
  - 10Gig-E
- Ceph Cluster
  - 4 Server, 14 HDDs each, 10Gig-E
  - Ceph 0.80.1 Firefly release
  - One server is icat + rs
  - irods pool + capabilities for client

Evaluation Timings

- Wall clock time of plugin functions

cluster_connect	73.67ms
create	72.61ms
Unlink	77.81ms
stat	89.90ms

Evaluation Results

Demo Time!

Summary

- iRODS manages namespace + access rights
  - iRODS resource server is client to one rados pool
    - Full access to all objects
- Single file overhead
  - Cluster connect per agent instance
  - Metadata updates + checks
    - Set physical path + get stat data
- High (parallel) throughput
  - Multiple user sessions in parallel
  - Multiple files per user session
  - Multiple streams per file
- Multiple resource server heads for one ceph cluster
  - Composed Round Robin resource

Contact & Sources

- Current development:
  - https://github.com/meatz/irods_resource_plugin_rados

- Contact:

  Matthias Grawinkel

  grawinkel@uni-mainz.de

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

JG|U