Data Pipelines

This session largely focused on the straw problem (™) presented by Luke from QBI. Discussion naturally broke down into an ingress and an egress section:

- **Ingress:** The primary problem with ingress discussed was that of slow, monolithic import. The general consensus was that import needed to be broken up into finer pieces and **ideally** made restartable. Some manual effort on the part of the user, such as passing the target ID of the Plate which was already imported, seemed acceptable. (There was a side discussion on the issue of symlink confusion along with the possible need for a `bin/omero mv` command)
- **Egress**: The primary concern with egress seemed to be how to make access to the original files uploaded in OMERO.fs more manageable. Without the API (i.e. the straw), file permissions are lost. A fuse plugin may achieve the desired mapping but needs to be shown sufficiently performant. Alternatively, sysadmins would accept the burden of manually configuring well-known map-points (e.g. via SMB or NFS) in which case the security settings would be explicit.
- We avoided the object storage question as not solving any of the fundamental problems.

Web of OMEROs/Federation

- Michel suggested that fundamentally this amounts to defining “gates” between services. Josh pointed to the XMPP protocol and the use of “capabilities”.
- The very generic term “federation” can be broken down into a number of related pieces of functionality:
    - 1a. Single-sign on (SSO), i.e. the ability to have one service use another for authentication
    - 1b. Client-side federation, i.e. the ability to list multiple servers in a single web-UI
    - 2. Master/slave federation, i.e. the ability to have one server (the master) pull in data from another server (the slave) and display it in the same UI, similar to a symlink. (*This is a slight misuse of the term “master-slave”*)
    - 3. Integration, i.e. a more permanent version of 2 in which data is copied for a longer period of time into the master. This implies some local storage and may or may not imply a synchronization process.
    - 4. Federated queries in which a single request requires data from multiple sources and some planning takes place to try to optimize the lookup time. Usually this is only difficult/interesting if intersections (as opposed to just unions) are possible. There are (at least) two forms of federated queries:
        - homogenous in which all resources are the same, e.g. all OMEROs.
        - heterogenous, e.g. mixing OMERO data with http://www.ensembl.org/

- Some questions that might be asked of a public resource could potentially leak critical information. A determination would need to be made if feature vectors (“fingerprints”) are available for public searches.
- A more general question which wasn’t gone into in detail was whether or not there should be a central registry of OMEROs which are available for such distributed queries.

Horizontal Scaling

Chris:
Can make straw bigger/smaller, but not increase number
- Fundamentally, spinning up a 2nd server requires too much shared state
- Not federation, this is about getting multiple OMERO servers to talk to the same DB

Douglas:
Tried running a completely standalone read-only server using the same DB and data store as the live version
- Started with read-only permissions for the RO OMERO user on DB, gradually increased privileges on tables until second OMERO ran
- Discovered OMERO required too many permissions

Chris:
Can create read-only servers with batch update of the DB e.g. weekly using PG replication.
- When server needs updating break replication, change to read-write, update, re-establish replication (rsync changed OMERO data, allow transaction logs to update)
- Update/replication process is fragile, and usually requires manual input

Chris:
Multiple Blitz is possible, but problem is session synchronisation.
- Can work if there are limited updates e.g. old JCB, but in that case it was just for rolling updates (so only one Blitz with active sessions at a time).

Douglas:
UX/reactivity is important, so may be better to throttle number of imports instead of having too many concurrent slow imports.
- Showing progression slowly vs waiting and doing everything in one go quickly
- Could part of import process be done offline by client, e.g. scanning and pre-preparing metadata?

Chris:
Problem with large filesets is it's limited more by the Hibernate and PG transactions than the scanning and metadata extraction.

- Most time is spent in the Hibernate ORM serialisation
- Breaking imports into smaller chunks should help, but loose consistency between overall import

Basu:
If multiple systems talking to the same DB with multiple imports could at least split total ORM work over multiple systems

Michel(?):
Any tools for checking whether an import is ongoing vs the server is frozen?
- OMERO user tools for checking/monitoring would be useful

Chris:
We've got access to the information in OMERO, but it's not presented

Douglas:
Profiling of OMERO?

Chris:
Downside of splitting work into chunks is more overhead, e.g. setup work for each chunk

Luke:
What are the current options when we run into scaling problems?

Douglas:
Looked into optimising file access with Isilon(?), linking up permissions in Isilon with those in OMERO using Isilon API so data analysts have direct access, and mounting FS on all nodes. Then IT said it wasn't possible, probably due to licensing costs of directly mounting Isilon FS.

Chris:
All high-throughput systems: analysts never read planes from OMERO, only limited metadata, all other file access is direct

Douglas:
Can't have a solution that makes more work to the analysis pipeline, any additional tools need to augment the pipeline, not get in the way of it
- Applied for grant to deal with the scaling problem

Luke:
+1 Douglas, running into the same issues

Chris:
Often recommends operational setups for OMERO HT analysis in GS

- but on open-source side infrastructure often limited by institution IT
- open-source doesn't have any funding for developing scaling
- GS instructs customers to have an infrastructure setup that works around the scaling problem
- open-source users don't have money to pay for infrastructure or GS

Luke:
Planning to implement a similar system to Douglas, give users direct access to files and ensure system file permissions are correct so they can only analyse their own data
- read only

Web apps

- Want to add plugins just for admins (not visible to others)
- REST api needs to support authentication (login) returning session key
- REST api for getting pixel data from within plane/shape and writing to OMERO tables
- Bug in OMERO.figure “Add Images” - ‘paste’ via menu doesn’t enable the Add Images button
- Did demo of ”Open with” PR #4630 which was well received
- Want to connect to 2 servers from OMERO.web framework
- Discussed stateless nature of http requests vv stateful Rendering Engine etc. and loading of multiple planes per request as strategy to load whole stack faster

Docker

Make concept of distributed system more clear to users, people are not aware OMERO.web can be deployed on a different host then server. This is particularly useful for setting up internal and external instances. External instance for public data only.