University of Washington's upgrade to 2.3

This is a summary of the University of Washington's experience migrating from Grouper 2.1 to Grouper 2.3. As you know the principal difficulty in this upgrade is the database conversion (26 to 30, to be exact). The present method parallels out previous conversion.

Some requirements of the upgrade

No interruption of service

Our group service (GWS), a RESTful API using Grouper's API as a back-end, is a core service at the UW. It is involved in a great many computer-computer and user-computer interactions---all day and night. We se about 200,000 - 300,000 queries per day, mid-quarter. More than that around the quarter breaks. Even during the night there are no hours with fewer than several thousand hits. Maybe 5% of these are updates.

Our REST API predates Grouper.

So a first requirement was that this service is not to be interrupted.

New database

We took the opportunity to migrate to a new database on a new host: Postgres from 9.1 to 9.6.4.

Steps in the upgrade

Update software

There were quite a few API changes going from 2.1 to 2.3, especially relating to attributes. We have some axillary processes that query some of Grouper's tables and these needed to be updated as well. Fortunately, these could be debugged on trial versions of the 2.3 database.

Update database

This was the bigger problem. Although the documented conversion steps basically worked:

1. One step of the conversion, updates of audit record keys, takes a full day. We have 40-50M audit records. We deferred this task.
2. Removing bad memberships after the conversion also took a long time–8 hours. There were 250K of them. Possibly another time we might consider dropping some indexes during this part.. This time we just accepted the delay.

Use of independent, synchronized group systems

We used independent, synchronized group systems three years ago. It worked then and worked this time. Because our GWS service is RESTful the only things that need to be sent from one system to the other are resource representations. There are some difficulties to be solved before such a cluster could be put into general service, but even without those this method can provide an easy upgrade path:

1. We have our existing group service (call it 'gws2') that uses Grouper 2.1 as a backend.
2. Create a new group service (call it 'gws23') that uses Grouper 2.3 as a backend. This was first deployed on a separate host, but finally deployed on the production application hosts in a rolling upgrade.
3. Set up an AWS FIFO message queue ('aws12') for sending updates from gws2 to gws23
4. Set up an AWS FIFO message queue ('aws21') for sending updates from gws23 to gws2. This part allows us to roll back the migration.
5. Start the aws12 sender. This will save all updates in the WAS queue. Lifetime there is 4 days. Should be more than enough.
6. Dump the gws2 database and load it into gws23.
7. Run the conversion scripts and the bad membership remover. This took a couple of days. It could have been one very long day, but I wanted to watch some of it run and also sleep at night.
8. Run the aws12 receiver to bring the gws23 database up to date with gws2. At this point there were a little over 3000 messages in the queue.
9. Start the aws21 receiver to copy updates on the new system to the old.
10. Switch two of the four application hosts to the new database (gws23) and switch DNS from the old to the new. This puts users on the new service, while keeping the old service up-to-date in case of any need to roll back.
11. Upgrade the other two hosts.
12. At this point users are on the new service with the upgraded database and never lost so much as a connection.

Summary

Our upgrade pretty much went as planned.

The cluster approach shows promise as a means to a very scalable and resilient groups service. We'll be looking into this.