GNU social - Optimizations on Load Balance System and Storage Usage

Miguel Dantas

```
Name: Miguel António Dantas
Email: biodantasgs@gmail.com
Project Name: GNU social
```

Summary

GNU social is a communication software used in federated social networks.

This project will:

- Improvements on the Federated Requests Queue (which will be shared between OStatus and ActivityPub)
- Improvements of the OEmbed plugin
- Improvements on the Image Systems
- Temporary posts
- Removing posts with no engagement
- Implementation of a circuit breaker

Benefits

A more lightweight Request Queue implementation using Redis. A more stable OEmbed plugin. Storage optimizations involving images and posts with no interaction. Reduction of resources uses in case of database error.

Deliverables

Currently, GS uses the database to store it's Request Queue. I will add a config option that will allow the use of a Redis cache to store the requests queues, instead of the database. This would fallback to the existing implementation if Redis isn't available. This could be detected during the installation and later selected through a config option. postActiv already implements this behaviour, however they implemented the Redis interactions directly instead of using the `predis` library. chimo's gs-rediscache plugin is a work in progress plugin which aims to use a redis instance as a cache. It uses the predis for this interaction. While it is not a queue, it seems natural to integrate it into GS and adapt into it the desired queueing system.

The OEmbed plugin requires some optimizations, since media (avatars, attachments, OEmbed/Open Graph media) is increasingly putting a strain on instances with a lot of notices. To tackle this problem, dansup has proposed a plugin that uses `JpegOptim`, `Optipng`, `Pngquant 2`, `SVGO`, and `Gifsicle` image optimizers and a new command that can be run periodically to optimize file size and garbage collect expired content (old avatars, ect). This command could further be used to optimize thumbnails. Currently GS stores the original file at full quality. I propose we convert thumbnails to jpeg and downsample them to the biggest size we need to generate all thumbnails. Thumbnail generation would be done as required, instead of upfront, by downsampling the bigger size. Further, we could store the original link, instead of the image and, in case a bigger thumbnail is necessary, attempt retrieving it. Periodically, unnecessary thumbnails would be removed. Additionaly, we could allow these thumbnails to be fetched by other instances. A config option could be added to determine if the original image or the link should be stored, the biggest thumbnail size to keep, the jpeg quality parameter. The migration from the old to the new system would be handled in the upgrade script and could be implemented by simply dropping all thumbnails, converting the existing original image according to the parameters and letting the new thumbnails be generated as needed.

OEmbed currently has some bugs, which will be diagnosed and fixed.

Support for temporary posts will be added. This will be achieved by prompting the user for a duration, when creating a post. Then, when the post is CREATED, a DELETE action will be sent into the resquest queue, along with the

duration. When the duration is elapsed, the DELETE action will be issued, thus deleting the post. Ultimately, there is no way to guarantee that the other instance will respect the user's wishes and actually delete the post, so no attempt is made to try to enforce it and this is done as a best effort.

Since many people host GS instances out of pocket, there's a growing concern towards the amount of resources GS can consume, especially in regards to disk usage. Therefore, I propose we automatically remove posts with no interactions, that is repeats, likes/favorites or replies. This behaviour would, naturally be controlled with a config option.

A circuit breaker is a software pattern wherein a connection is attempted, but if it fails, a timer is started, to avoid spending too many compute resources attempting a connection. This pattern could be implemented in the database connection. It could, in theory, be implemented in the HTTP connections, however this shouldn't be necessary, as all requests will be handled by the queue, which serves many of the same purposes.

Plan and timeline

May 6 - May 27

- Familiarize myself with the community
- Familiarize myself with the code (further insights in):
  - Image handling
  - Cache related plugins
  - Queue related plugins
  - Database access library

May 27 - June 24

- Improvements on queues
- New image handling system

June 24 - June 28

- First evaluation

June 25 - July 22

- OEmbed optimizations
- Support for temporary posts

July 22 - July 26

- Second evaluation

July 23 - August 19

- Circuit breaker
- Random tasks

August 19 - 26 August

- General review
- Writing tech report
- Final merge
- Code submission and final evaluation

Communication

My code will be hosted on a fork of Diogo's fork. This will be done to prevent accidental commits, as a significant part of the community has switched to Diogo's repo. Further, communication will be made through IRC, on the `#social` channel on freenode, where my nickname is `biodan`. I will communicate my progress at least twice a week, besides asking for clarifications.

Qualification

GNU social is a project in which I am truly interested in getting involved in and contribute to. I really find the concept of federated social network amazing and being able to work on such a project seems to me like an excellent opportunity.

I am rather expirienced with Bash, as well as some experience with PHP, SQL and Web Frontend Development (HTML5, CSS3 and JavaScript). Furthermore, I am currently enrolled in my second year of Computer Science in the Faculty of Sciences of the University of Porto, Porto (Portugal).

For working in GNU social, I will have to learn about the project itself and further familiarize myself with PHP.