1. Introduction

During the month of September, the Sakai OAE project team started to execute on a plan that recommended the retirement of the existing back-end technology, Nakamura, and its associated technical debt, performance and production problems. A milestone was set at the end of 2012 that would evaluate whether or not a solid technical foundation has been put in place that allows for Sakai OAE to be usable at a much higher scale as a multi-tenant SaaS and cloud-compatible platform, allowing for faster feature development at the same time.

This report is the third and final progress report summarizing the progress that has been made against the end-of-year milestone. The first two reports can be found at [1] and [2], describing the development of a new back-end using Node.js, Cassandra, ElasticSearch, etc. and the performance testing that has been done on that platform.


2. Architecture update

The technology stack as outlined in the previous progress reports remains almost completely intact. However, three related modifications are worth noting. It is now possible to designate app servers as activity processors, preview processors and indexing nodes. All of these are regular app servers using the same Hilary code base, which can be designated for these activities, but can also be combined in small scale deployments.

- **Activities:** One or more app servers can be designated as the activity processors. Activities are user triggered actions like uploading a new file, commenting on a piece of content, sharing content, creating a new group, etc. that generate an activity on the activity wall for all those that the activity is relevant to. The activity processors are regular app servers, which should not be accessible through Nginx in a large deployment in order not to have them serve REST requests at the same time. When an activity occurs, a task will be sent to a RabbitMQ task queue, where it will be picked up and processed by one of the activity processors. If you add more activity nodes to serve the task queue, it is expected that the performance of activity processing should improve linearly. It is also worth noting that processing an activity is an asynchronous process that happens outside of the main request cycle, which prevents unnecessary delay in the user's perceived response time.

- **Preview processor:** The Preview Processor has been rewritten in Node.JS to reduce the number of different languages used in the project, as well as leveraging some existing Hilary code. When a file gets uploaded, a task is sent off to a RabbitMQ task queue where it will then be routed to a preview processor node. RabbitMQ takes care of the routing and supports the horizontal scaling use case. To generate the actual preview images, LibreOffice is used to convert Office files, pdftk is used to split PDFs and GraphicsMagick is used to crop and/or resize images. One or more app servers can be dedicated as a preview processing nodes will be necessary, allowing for linearly improving the preview processing throughput.

- **Search indexing:** One or more app servers can be designated as search indexing nodes. When a new entity requires indexing, a task will be sent to a RabbitMQ task queue, where it will be picked up and indexed by one of the search indexing nodes into ElasticSearch. If you add more search indexing nodes, it is expected that the throughput should improve linearly.
3. Progress

At the time of the previous progress report, a number of large foundational areas of functionality (i.e. search and file uploads) were under review. These have now been reviewed, merged, tested and integrated into the UI and the Model Loader. Once this was done, the bulk of the work focused on re-enabling and re-integrating the UI as well as additional feature work.

3.1. Search

The ElasticSearch based implementation [1] of search functionality has gone through various rounds of review and refinement, as well as additional performance testing, and is now part of the main Hilary codebase. Next to the OAE general search feed, additional feeds have been created for searching through libraries, memberships lists and member lists, as well as less intensive search feeds that can be used for all autosuggest searches.

The initial implementation would only return results when a full match existed, and various rounds of search quality tuning have been done, improving search quality considerably. There will probably be a need for additional rounds of tuning to find the right cut-off point as we are currently seeing too many results returned, but the relevance sorting does seem to be behaving appropriately.

A privacy strategy that determines when a result will be returned and when it will not be returned, based on the visibility settings of the entity as well as whether or not the search is cross-tenant or not, has been drafted and implemented [2].

[1] https://confluence.sakaiproject.org/display/3AK/Hilary+Search
[2] https://confluence.sakaiproject.org/display/3AK/Visibility+Matrices

3.2. Profile pictures

Uploading and cropping profile pictures for users and groups has been re-implemented. Rather than replicating the custom image cropping functionality available in Nakamura, we now rely on GraphicsMagick to perform the image cropping. When a selected area is cropped out, it is resized in two different sizes, a small size for profile picture icons, and a larger one that can be used in the search or activity results.

Profile pictures have been wired into the different list views in the UI, and have been hooked up to the profile picture overlay.
3.3. Content comments

Commenting on content items has been re-implemented, including the addition of being able to reply to comments on multiple levels. A new content comments widget has been created that allows for creating comments, seeing all existing comments and deleting comments when allowed, which has been hooked into the content profile page.

3.4. UI Integration

The bulk of the work since the last progress report took place in the User Interface, as there was no working UI for Hilary yet. In the first instance, the UI has been hooked up to the new back-end services exposed by Hilary. In doing that, the team has attempted to simplify and clear some of the technical debt that had been introduced over time where possible.

In a few areas, we have opted to leapfrog to new design work straight away, like content comments, the library page, search pages, memberships pages and members pages. In other areas, we have chosen to re-implement the existing Sakai OAE UI, often simplified.

3.4.1. API improvements

The Sakai OAE Front-End APIs have been reworked completely to use the new Hilary back-end services. In doing this, considerable simplifications and improvements have been made due to the fact that there are now dedicated REST endpoints available for all actions, which should make the APIs a lot easier to use for widget developers as well.

The API outline has been completed for the existing Hilary functionality, and has undergone a few rounds of refinements. Most of the API functions have also been implemented, although a few still require implementation.

We have increased the usage of Require.js, which is now handling all of the dependency management and asynchronous loading of modules and plugins.
3.4.2. Internationalization

The i18n properties file parser used for reading the internationalization bundles has been updated from a homegrown properties parser to a 3rd party plug-in, which makes parsing the properties files a lot more robust and fully compliant with the standard.

Other than that, the internationalization and translation process has remained largely unchanged. As the UI has not dramatically changed yet, the existing French, Spanish and Russian translations can still be used.
3.4.3. Configuration

Additional back-end services have been written to make the publicly available configuration settings for a tenant available for use in the User Interface. This has now replaced the old config.json and config_custom.json files, as the UI has been hooked up to use this new feed.

When a configuration change is made in the admin UI, the change will immediately propagate to the configuration feed and be available in the UI. This means that there is no longer a need to rebuild and restart the system when making a configuration change. Configuration changes can be made on either a global level (all tenants) or on an individual tenant level.

![Configuration Interface](image1)

3.4.4. Landing page

The landing page has been re-enabled, including the footer and the top navigation. As no explore-like functionality (e.g. suggested content, categories, etc.) is available yet, the categories carousel and suggested content widget have been disabled. The recent activity stream has been disabled as well in anticipation of new landing page designs.

![Landing Page](image2)
Inside of the top navigation, the autosuggest search feature has been re-enabled.

The login pop-over has been extended to reflect the supported login strategies for the current tenant (local login, Twitter login, Google login, Facebook login, etc.).
3.4.5. Register page

The register page has been re-enabled. This will only be accessible for tenants that allow users to register themselves.

![Register page screenshot]

3.4.6. Activity page

When users log in, they will be taken to their personal activity page, listing all the activities relevant to that user, instead of the dashboard page we had before. This is in line with the new designs presented at the Atlanta conference, and all of the HTML and CSS required for this has been created. The widget has not been hooked up to the actual activity feed yet, as the back-end functionality is currently under review.

Longer term, the goal is to bring back some dashboard-like functionality on the right hand side of the page, but for now we have opted for the simplicity of just the activity wall, which subsumes all of the dashboard widgets that existed before.

Groups will have their own activity page as well, listing all the activities related to the group.

![Activity page screenshot]
3.4.7. Lists pages

The Library, Memberships and Group members widgets have all been re-implemented using the lists framework outlined in the designs presented at the Atlanta conference, which simplifies the UI quite a bit. Each of those lists have 3 different views available; grid view, list view and expanded list view, and profile pictures have been hooked up into the lists as well. Preview processed images have not yet been incorporated into the lists, which will be one of the next tasks.
Default list views are now all driven by feeds that read directly from Cassandra, rather than relying on search. This significantly unloads the search servers, which has a considerable impact on search performance. This also eliminates the indexing delay, making new items available in the lists straight away.

Searching inside of the different lists has been implemented and hooked up as well.
3.4.8. User profile

The user profile pages have been re-enabled. This has mostly been done as a demonstrator of the functionality though, as we expect heavy design changes in this area in the medium term future, including the investigation of integration with platforms such as Vivo.

3.4.9. Add content overlay

The add content overlay, which is used for uploading files, creating new documents (metadata only) and adding links has been re-enabled and considerably simplified. The ‘Add from existing’ functionality has been removed in anticipation of the new designs.

Unimplemented features like tagging have been temporarily disabled as well.
3.4.10. Content profile and comments

The content profile page has been partially re-implemented. For now, only images and links will have previews on the actual content profile page, and other content items will need to be downloaded until the preview processor has been fully re-integrated. Comments and replies to comments can be made at the bottom of the page.

Uploaded files can be downloaded, and content access can be managed by the content managers. Content can also be bookmarked or shared by non-managers.

Revision history and uploading a new version of a file have not been re-implemented yet, even though the back-end functionality is available. Showing and managing the content metadata hasn’t been re-enabled yet either.
3.4.11. Create group

The ‘Create group’ page has been re-enabled, although there is currently only 1 group type available. Other group types have not been implemented yet, in anticipation of future design work.

Groups can be created with a custom URL, visibility and membership, with the addition that it is now possible to make a group private and still make it possible for users to find the group and request to join it. Members can be added and managed as well.

3.4.12. Group profile

The group profile page has been partially re-enabled and dramatically simplified. For now, each group has an activity page, a library and a members list, in anticipation of the pinning concept being implemented. Members can be added and managed and a group profile picture can be set.

It is not yet possible to modify the group’s metadata. People can request to join a group or join the group (depending on the joinability settings), but this hasn’t been hooked up to the UI yet.
3.4.13. Search pages

The search all, search content, search people and search groups pages have all been re-implemented, using improved infinite scrolling and the new lists framework. The user and group profile pictures have been hooked into this as well.
3.5. Widget development

In addition to simplifying and consolidating the front-end APIs, a number of simplifications have also been made to the widget format and widget development process, and the existing widgets have been updated to comply with this.

Widget javascript files had a number of Sakai OAE specific statements that needed to be in there for a widget to load successfully. These have now been removed and replaced by standard Require.js functionality. The widget loader has been rewritten as well in order to make the loading more robust.

Distributing and installing widgets has been changed dramatically. Instead of the widget developer manually uploading the widget zip file and the deployer manually downloading the widget file and extracting it to the right directory, we are now using the Node Package Manager (NPM) to publish and install widgets. This means that a widget developer can either distribute the Github URL to their widget, or can push their widget up to NPM. The deployer can add this to the ‘package.json’ file and NPM will automatically make the widget available or download updates when desired. This also significantly reduces the patch that deployers need to maintain to just the ‘package.json’ file.

It is now also possible to distribute groups of widgets and keep local widgets grouped in a folder separated from the other Sakai OAE widgets.

3.6. Tenant privacy

In the Hilary back-end, the ‘tenant permeability’ concept has been baked in from the very beginning. The idea is that, if desired, different tenants can interact with each other, for example by being able to find content from other tenants, add people from other tenants to a group, etc.

Since the last progress report, quite a bit of work has happened around tenant privacy. First of all, it is now possible for a tenant to express whether or not they want to be part of the list of permeable tenants. If they decide not to, they will not be able to see and interact with people, groups and content on other tenants from the current tenant. When a tenant switches from being permeable to not being permeable, all cross-tenant links that have already been created will be retained and remain usable, but it will not be possible to create any new ones.

Some fixes have also been done to the permission side of things, where we now make sure that the ‘logged in only’ permission property only applies to users on the current tenant, and not to users on other tenants.
3.7. Activity

Activity feeds have been improved and the server-side implementation has been re-implemented. The new activity streams have been modeled after the JSON ActivityStreams 1.0 specification, providing a generic data-model that should make exchanging data with external systems easier. More information on the ActivityStreams specification can be found here: http://activitystrea.ms. The implementation is expected to be ready for code-review and performance testing before the end of this week.

Next to that, activity aggregation has been implemented as well. This is the process of collecting similar activities and distributing them to activity feeds as a single summarized activity (e.g., “John Doe shared 5 content items with Introduction to Mythology”).

The activity implementation also provides the ability to run a cluster of activity processing slaves. This prevents activity processing from negatively impacting user request latency, and the overall perceived performance of the system by the user. Similarly, an increase in user requests should not impact the amount of time it takes for an activity to appear in streams once it has occurred (delivery latency). As more activities are created by users, the delivery latency will naturally increase. Special care has been taken to ensure that the delivery latency can be improved linearly by adding more application nodes to the activity processing cluster. This assertion will be validated when the implementation is ready to undergo performance testing.

3.8. Content preview

The preview processor has been re-created in Node.js, and is currently under review. It will process all Office files, as well as PDFs and images, by generating thumbnails in various sizes (small, medium, large) for each of the pages in the document. This is done to ensure that we send the minimum amount of data required to the user’s browser.

Once the preview processor is merged into the Hilary code base, it needs to be hooked up to the content preview widget and the various lists.

3.9. UI Production build

There is currently an open pull request that re-enables the UI production build, allowing for the removal of a considerable amount of custom production build code in favor of Grunt.js. Even though the pull request for this is in, the team is still sorting out some final issues with this.

The Nginx config file has been modified to set non-expiring caching headers on all static resources and static batch gets, as well as cache-busting headers on all dynamic resources, which has been tested on the performance testing environment. Gzip has also been enabled in Nginx in order to reduce the size of the files sent to the browser.
4. Conclusions

Back in September, an end-of-year milestone was set that would determine whether or not the Sakai OAE project still had a future. The deliverable for that milestone was to evaluate whether or not a solid foundation could be provided that allowed for Sakai OAE to be usable at a much higher scale as a multi-tenant SaaS platform, allowing for faster feature development at the same time. Note that trying to re-implement all existing OAE functionality as-is as fast as possible was never a goal.

Along the way, the team would attempt to improve the widget development experience, simplify code and UI elements and leapfrog to new design work where possible.

In terms of functionality, the following table provides an overview of the features that have and have not been re-implemented since:

<table>
<thead>
<tr>
<th>Implemented</th>
<th>Not implemented (yet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permissions</td>
<td>Tagging</td>
</tr>
<tr>
<td>Users and user management</td>
<td>Collections</td>
</tr>
<tr>
<td>Groups and group management</td>
<td>Sakai Documents</td>
</tr>
<tr>
<td>Content</td>
<td>Contacts/Following</td>
</tr>
<tr>
<td>Search</td>
<td>Messaging</td>
</tr>
<tr>
<td>Activity</td>
<td>Public browse experience</td>
</tr>
<tr>
<td>Multi-tenancy</td>
<td>Related content</td>
</tr>
<tr>
<td>Network (tenant permeability)</td>
<td>Group templates and areas</td>
</tr>
<tr>
<td>Content previews</td>
<td></td>
</tr>
</tbody>
</table>

Next to feature development, there has been continuous performance testing along the way. This has shown that the architecture is capable of horizontal linear scaling. The team has set up and maintained a reference environment in the Joyent cloud, which has shown that a large scale deployment could be run at a reasonable cost. Next to that, a high level of code coverage and unit testing has been maintained throughout the process.

We believe it is fair to say that the milestone has been achieved, and technical foundations have been put in place that could be used to continue to evolve this into a viable product. Most of the foundational pieces (permissions, user and group management, file uploads, search, etc.) are now in place and sufficiently tested, which allows for focus on feature development.
5. Next steps

In the first months of 2013, the project team and stakeholders will be spearheading a conversation around the (initial) strategic direction for the project, as well as a contributing partner model, as both of these will be critically important to pick the right set of short, medium and long-term priorities and determine a development and design roadmap. Early conversations have moved in the direction of continuing to push the collaborative aspects of Sakai OAE, but pushing the teaching angle and pushing the repository angle have been mentioned as options as well.

Whilst this conversation is taking place, the team will continue to work on a number of items that need doing irrespective of the strategic direction:

- Solidify and complete all of the UI elements related to the areas of functionality listed in the implemented column. This includes re-enabling the revision history overlay, content metadata, related content, group join requests, etc.

- Continue to move toward the new designs presented at the Atlanta conference, including better context switching, list improvements, full-width application, better add content overlays, etc. Combined with the previous goal, the team will be working towards a UI containing all of the implemented functionalities, and ready for testing and production use, albeit with the reduced scope.

- **Update the performance tests to exactly reflect the HTTP requests done by the browser, before and after caching.**

- Continued development around Single Sign On integration.

- Present the new project direction and progress at the Paris EuroSakai conference.