SynBioHub DBLink Plugins

By: Sean Nemtzow, Anirudh Watturkar, Rahul Varki, Liam Murray

Project Motivation

The goal of this project was to connect SynBioHub to both the GenBank and UniProt databases through SynBioHub's new plugin functionality. Accurately being able to link SynBioHub component pages to corresponding information stored in the GenBank and UniProt databases would allow for more efficient design-oriented research which would enhance the usefulness of SynBioHub. We envisioned that a user would be able to search for a part of interest on SynBioHub and through the use of our plugins would seamlessly be able to view and filter relevant GenBank and UniProt information for that part. We got the idea for this project after communicating with professor Chris Myers at the University of Colorado Boulder, a main contributor to SynBioHub. We were primarily in communication with Jet Mante, a graduate student in the Myer's Lab, who provided us project guidance throughout the semester.

SynBioHub Connectivity

There are three endpoints which must be included for any SynBioHub plugin: status, evaluate, and run. Status returns the status of the plugin (i.e.; whether the plugin is running or not); evaluate is used to determine whether the plugin is applicable to the SynBioHub page currently being viewed. Lastly, run is used to retrieve information from SynBioHub and send information to the html file.

UniProt Plugin

Java API:

The UniProt Java API allows for BLAST searches of the UniProt database using amino acid sequences as queries. Since SynBioHub stores nucleotide sequences, translation is performed using Biopython's Seq object. This protein sequence is then queried through the Java API BLAST search, and the accessions for the first fifteen hits are retrieved.

RESTful API:

The accessions retrieved from the BLAST search are then queried using the Protein REST API. This API provides JSON formatted objects which are parsed for the specific information desired for inclusion in the plugin, and these data are saved in a new JSON object before being provided to the corresponding html file.

Genbank Plugin

Plugin Server:

The plugin server hosts a RESTful API to interface with the main SynBioHub instance. It informs SynBioHub that it only accepts “component” types for evaluation. When a run command is called, it sends a FASTA formatted file to the AWS cluster Entry Node to process the BLAST search. It also renders the HTML page and provides it the information to check the communication server for available data.

Communication Server:

The communication server is the entry point for the AWS database cluster. It is the communication bridge between the parallelized BLAST searches and the SynBioHub plugin. It receives the FASTA file mentioned above from the plugin server via an http POST request, and responds with an ID that represents the plugin’s request. It then sends this FASTA file to each database server via POST request, and awaits the responses. Once all database servers respond, the communication server filters the top ten results, gets the GenBank file data for each result, formats and stores it, then waits for the HTML file to request the data. The server then caches the final results in a local directory for quick lookup later. As default settings, the server caches the last 100 query results.

Database Server:

Each database node in the AWS cluster houses a small section of the Genbank database, which we call a fragment. The database server runs on the database node and communicates with the entry point node. When it receives a FASTA file from the entry point node, it mounts the database fragment inside a docker container and runs a

BLAST search inside the container. When the results are done, it sends the top ten matches back to the entry point node.

HTML/Javascript

The javascript and HTML is given a query ID and the IP address of the communication server by the plugin server and uses the ID to check on the status of the BLAST search. Every two seconds, it queries the entrypoint node in the AWS cluster to check on the results. When they are ready, it downloads the results and populates the dropdown menus and tables. Additionally, it provides numerical fields to filter the results based on the quality of the match, as well as hyperlinks to cross-referenced databases if the user wants to learn more about a result.

Special Instructions for Compilation

The core functionality of this project relies on having a SynBioHub instance up and running. Please be sure to carefully follow all of the steps in the SynBioHub wiki for installation and setup. These steps are also documented in the README, but are in greater depth on the wiki. In order to test that the plugins are working, you will need to upload some components onto your own instance. The easiest way to do this is to download a .gff file from SynBioHub and upload it to your personal instance via the submit button on the SynBioHub page. An example of one we've used is this part.

Also, please be sure to disable any firewall protections if the plugin does not seem to be working, as firewalls may block incoming/outgoing requests to external IP addresses, which is necessary for the plugins.

Please note that for the GenBank plugin, downloading and extracting the database takes a long time, and especially for testing, we recommend you only download a small subset of the BLAST database, as this will reduce the amount of instances necessary and the amount of time it takes to download each database fragment (**note: always download nt.00 in addition to any other database partition, as it has required information for other database fragments**). Also, to reduce the run time for the GenBank plugin, please make an API key for Entrez as described here.