10/26/2017

Technology Feasibility Report

Team U.I. Fit

Team Members:
Charles Chatwin
Matthew Burns
Tanner Brelje
Joshua Gutman

Sponsor: Dr. Abolfazl Razi
Faculty Mentor: Dr. Abolfazl Razi

The purpose of this document is to provide an analysis of possible technological choices for the BioNetFit project.

TABLE OF CONTENTS

INTRODUCTION

The field of molecular biology is an ever advancing science that requires tedious experimentation and research. In order to generate concrete research results, many molecular biologists must place precious time and resources into large-scale experiments. These experiments are used to show the process of biochemical molecular interaction, or in layman's terms, the result of the reaction between two or more molecules. In order to aid this meticulous process, Northern Arizona University graduate Brandon Thomas, in tandem with an experienced team of graduate students and molecular biologists, created the program BioNetFit. BioNetFit is a command line tool that was created to provide a fast and easy method to simulate complex molecular bonds. These simulations allow researchers to later run the tests in a real environment in a way that gives them a degree of confidence in the expected interaction.

Built on top of BioNetGen and NFSim, BioNetFit allows researchers to simulate a single experiment many times with a range of parameters. The results of each of these simulations is compared to an experimental results file that the researchers upload. Because the combinations of different parameters scale non-linearly, various methods are used to select the best combinations of parameters, including genetic algorithms, simulated annealing, and a few others. After the simulation has been run many times (usually thousands), BioNetFit creates a final output file that shows information about the molecules involved in the reaction.

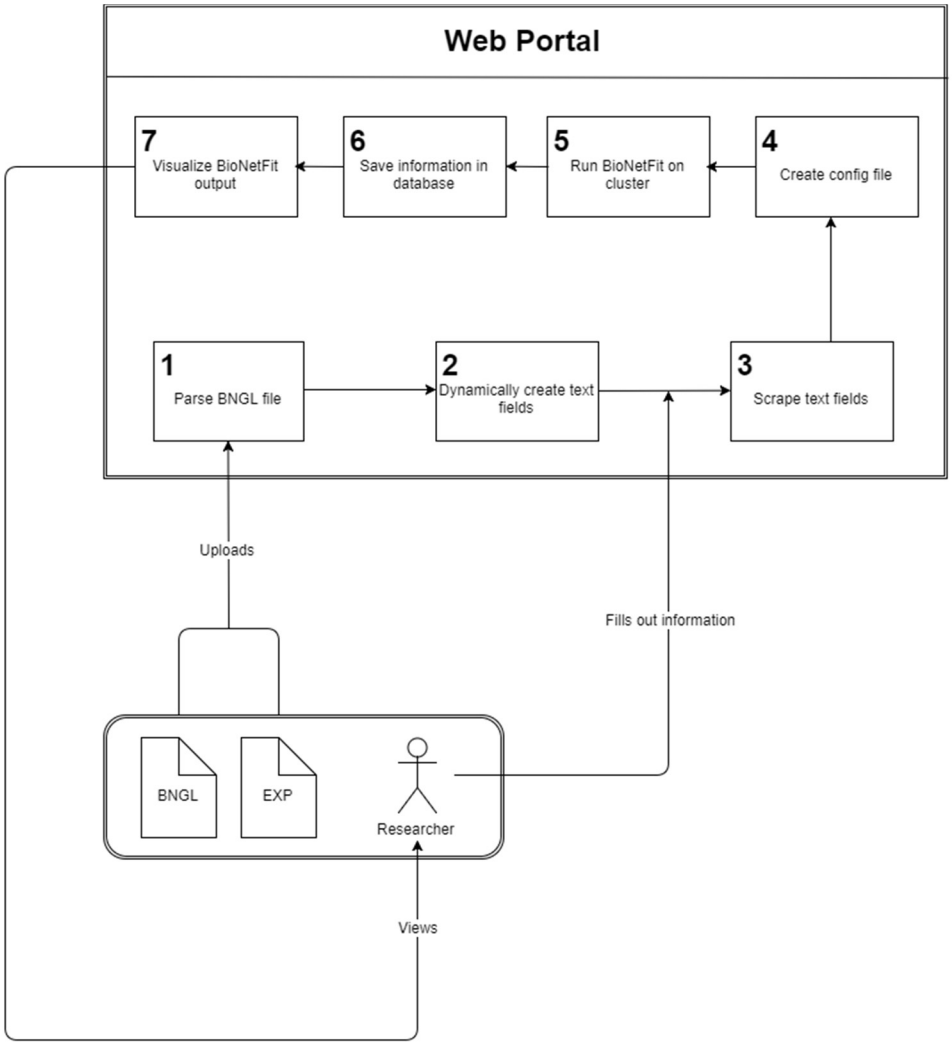
While the program is incredibly useful, its implementation is not user friendly. It exists as a unix-only, command line tool. Researchers who are not technically proficient will struggle at getting the program to run and will attempt to find other avenues when faced with having to spend time learning a new system to run their tests. Additionally, the results currently output by the program are difficult to digest, and do not lend themselves easily to analysis for data collection by researchers. Without much labeling or clear direction in the results, researchers will have to extract results from a command line output, which as previously discussed, is problematic for many scientists. Lastly, the program cannot run large scale experiments in a short amount of time. This lessens the practicality of the program as the experiment becomes more and more grand in scale. In order to make BioNetFit the stellar application it can be, these issues need to be addressed. If done correctly, the program could see widespread use in molecular biology labs all over the world.

Created in order to tackle these challenges, Team U.I. Fit, composed of Charles Chatwin, Matthew Burns, Tanner Brelje and Josh Gutman, will develop software solutions in order to turn BioNetFit into the powerful program it can be. Led by client and mentor Dr. Abolfazl Razi, our team will aim to:

- Create an attractive and simple web portal for BioNetFit.
- Visualize the results of BioNetFit into easily digestible data for researchers.
- Implement parallelization in order to run large experiments on a computer cluster.

Firstly, our team will develop a simple and effective Web 2.0 Graphical User Interface that will house the BioNetFit software. This web page will be easily accessible and easily usable by any researcher who wishes to run tests using the BioNetFit software. In the web portal, the user can either create their own BNGL file from scratch, or upload one that was either previously downloaded, or was saved remotely on the website itself. From

here, the user can run the file either locally on their own system or, in the case of a large file, run the file on the computing cluster Monsoon, available on the Norther Arizona University campus. From there, the program will output a configuration file that will be visualized to the user using graphs and charts. Finally, the user can either save and visualize the outputs on the website, or download them to their own local machine. The BNGL files can then be tweaked if wanted, and the experiment can be run again.



In this document our team will go over the project and discuss pitfalls and solutions that we believe it presents to us and what we are preparing to work around them. We begin in the Challenges section, where we have an overview of the individual problems facing us so that the reader has a better understanding of the moving parts of the project. Then we will analyze each area individually in the Analysis section, presenting possible solutions and workarounds before sharing which solution we are preparing to move forward with, and the reasons behind our team's decision. After that we will provide our big picture; how our team envisions the systems described in the previous sections coming together and operate as a whole

CHALLENGES

Our project has several challenges that it will need to overcome. We will be dealing with a web portal, so we will need to have a language that can interact with this web portal type of interface. We will also be storing information through the web, thus we will need the use of a database to store information online. As we are dealing with the web, we are also going to be accessing information and using programs that will result in us needing to use SSH (Secure Shell) to use private information over a private service. We are also needing to look into the feasibility of using command line commands to run specific programs to make using our website easy and user friendly (allowing us to have a local machine execution). We also need to integrate HPC (High Performance Computing) with a cluster to allow the use of a high computing cluster to quickly solve very large and resource/time-consuming projects of large scale. In order to integrate HPC, we will possibly need to troubleshoot the original program to efficiently use MPI (Message Passing Interface). The goal of the Message Passing Interface is to establish a portable, efficient, and flexible standard for message passing that will be widely used for writing message passing programs. MPI is the first standardized, vendor independent, message passing library. The advantages of developing message passing software using MPI closely match the design goals of portability, efficiency, and flexibility. These advantages are important to the BioNetFit program. So, our challenges include:

- We will need a language that works well with the creation of our web-based project.
- We will need a reliable data store backing our product that is capable of secure, highly-accessible large storage for data.
- We will need a secure interaction and user authentication for our web portal.
- We will need a way to have local machine execution from the web.
- We will need to integrate HPC with the cluster as well, which may require troubleshooting of the original program and MPI.
- Possible troubleshooting of the MPI integration with BioNetGen to optimize its efficiency.

ANALYSIS

The following is an analysis of the technology choices for the assignment, as well as the preferred methods for each one.

HTML GENERATION

The user will start using the website by uploading a BNGL file. The website is then responsible for reading the BNGL file and creating an HTML page where the user can enter in values that correspond to keywords in the uploaded file.

1. Using Javascript(JS) and PHP

   In this method we would keep the web portal side limited to JS, PHP and HTML, and would use JS to run a PHP script that would take in the uploaded file and then scan it for pre-defined keywords to generate the HTML boxes.

   Pros:
   - A more "Unified" solution, it allows for the site to be easier to create as the team will focus more tightly on already used languages.

   Cons:
   - Cumbersome. The functions needed to do this in PHP and JS ended up being complex and cluttered up the webpages they existed on.
   - Since we believe our web portal will be both client and server side, JS is a client side language and it's a challenge to bridge that gap.

   Testing:
   - The testing done with this approach was not able to get to the beefier parts of the program such as scraping for keywords or generating boxes before it passed the level of complexity that we were looking for, and was deemed as an unfavorable solution.

2. Python via Django module

Using the third-party open-source Django module, HTML pages can be generated by having a template, and specifying where the dynamic content occurs in the template. The dynamic content itself can be defined as

strings in the program, and can be inserted when certain keywords appear in the uploaded file. The uploaded file would be parsed in vanilla Python.

Pros:

- Django is easy to use and well-documented
- Parsing files in Python is trivial
- Maintainable -- if new features are added to BNGL, it would be simple to account for those in the program.
- Can interface with MongoDB directly

Cons:

- Third-party dependency
- Requires AJAX to be able to run Python scripts from Javascript

Testing:

- A web project was created, and an HTML page was able to be generated using Django and user-input using just a few lines of code.

3. Python via Flask module

Flask is another third-party Python module. A competitor to Django, it has many similar features. Flask is newer, and has split the Python community in half. Like Django, it can use Python as an interface to dynamically generate HTML.

Pros:

- Flask is simple and flexible
- Parsing files in Python is trivial
- Maintainable -- if new features are added to BNGL, it would be simple to account for those in the program.

Cons:

- Third-party dependency
- Requires AJAX to be able to run Python scripts from Javascript
- MongoDB compatibility has only been tested on outdated versions of Python
- Fairly new -- less information online than alternatives

Testing:

- An HTML page was dynamically generated using Flask and user-input.

Other options were looked over and thrown out of the possibilities to be considered of languages for other reasons. *Dreamweaver*, a program to design web pages, was not considered for reasons such as not being able to generate dynamic code. Using *Perl* language was thrown out because it is very outdated and is not something anyone on the team has in depth knowledge of. *C, C++, and C#* were not used as a language due to accessibility and ease of access reasons; thus they were not considered. We did not need to continue looking too far into other options because we found an option that was very popular for these things and more recent.

Decision

We will be using Python to parse the BNGL files, and Django to dynamically generate HTML pages. The simplicity of Python means an easier and less error-prone development, as well as a more maintainable and scalable platform. Because Django can interface with MongoDB, our Database of choice, a more unified development environment is possible. Although AJAX is needed to run the Python script, its use is minimal and will also allow for concurrent calls to the Python script in the event that multiple users are using the site at the same time.

	Dynamic HTML gen.	**Simple**	**No third-party dependency**	**MongoDB interface**	**Ample documentation**	**Text-field scraping**
JS and PHP	✓	X	✓	X	✓	✓
Django	✓	✓	X	✓	✓	✓
Flask	✓	✓	X	X	X	✓

DATABASE SYSTEMS

In order to access the BioNetFit GUI, the user will first have to create an account with the website. This login would consist of a username and password, and possibly an email address in order to allow for user confirmation in the case of forgotten passwords. In addition, the molecular combinations done by the user should be saved by the website based on the user account. This would allow users to re-access formulas they had been previously working on, eliminating the need for saving the document on the client side. In order to save all of this information, the website is going to require a database. This database must be large enough to hold all of the login data, and possibly many different files. Additionally, it must hold the data used for the HPC access of the cluster. This data must be secure, and the database will likely have to support encryption protocols in order to assure this.

1. MySQL

   Our first of two solutions to this problem would be MySQL, by the Oracle Corporation. This would allow us to implement a simple database structure in order to catalog the data of the users.

   Pros:
   - The program is much simpler than its contemporaries, making it easy for our team to implement the database.
   - The program is currently taught at Northern Arizona university, allowing for easy assistance from professors and previous knowledge of the system.
   - Implementation, as well as upkeep, will be easy to do and maintain.

   Cons:
   - Since its acquisition by Oracle corporation, the program is no longer open source or completely free for public use, making it a difficult option for student use.
   - The program can handle the login portion of the website, but may have trouble holding the data for a large amount of files.
   - Not easily compatible with Python.

   Testing:
   - The program was able to hold the login information for multiple users. However, we were unable to test how the server would handle saving large files and being accessed by a cluster. Previous uses of this program however point to this being a weak possibility.

2. MongoDB

Our second of two solutions is MongoDB, a free, open source program that prides itself on being the number

one choice in big data storage. While more complex than MySQL, the database system is more intricate and easier to adapt to the system.

Pros:

- The database system is more complex, allowing for modifications to fit the needs of the program.
- The program is widely used by a majority of major corporations, allowing for many implementation examples to learn from.
- Implementation, as well as upkeep, will be challenging but rewarding as the system is easier to scale.
- Easily compatible with Python through PyMongo.

Cons:

- The program has a steep learning curve, making it hard to learn for novices.
- The system may be more than what is required for the project, which could possibly affect the company's data storage.

Testing:

- Like MySQL, the database could handle simple login information. Also like our MySQL testing, we were unable to test the file storage capability. However in this instance, the other companies that use this database are large and work with giant amounts of data. Because of this, we can infer the system will hold the files from the GUI.

Decision

Our final decision for databasing is MongoDB. It provides the exact qualifications needed for the server, as well as the space needed to store as the multiple files created by the users. The one major drawback is the learning curve, as many of our team members are not familiar with databases, especially higher level ones like MongoDB. Given enough time however, the program should be mastered by one member of the group and will become common to the remainder. Overall, the software should serve its purpose well.

	Simple	**Maintainable**	**Scalable**	**Python compatible**	**Open-source**
MySQL	✓	✓	X	X	X
MongoDB	X	✓	✓	✓	✓

SSH

Once all the necessary files have been generated, it would be convenient for the user to be able to run BioNetFit on a cluster or server of their choice. This can be accomplished by using SSH to transfer the files and then run BioNetFit.

1. Python via Paramiko module

   A third-party open-source library called Paramiko allows Python to establish SSH connections to transfer files and run commands remotely with just a few lines of code.

   Pros:
   - Very simple to use
   - Has functions that fully encapsulate both file transfer and command execution
   - Connections can be kept open for long amounts of time

   Cons:
   - Third-party dependency
   - Must run on server instead of user's local browser
     - Prevents the server from SSHing into clusters or servers that require a VPN to access
     - Possible security risks with allowing users to SSH into arbitrary host
   - Requires AJAX to call Python script from Javascript
   - Possible security risks when using user's SSH username and password

   Testing:
   - An SSH connection was established between a team member's laptop and Monsoon.
   - Commands were executed on Monsoon, and the output of those commands could be read on the laptop.

2. Node.js via ssh2 module

A third-party open-source library called ssh2 allows node.js to establish SSH connections to transfer files and run commands remotely.

   Pros:
   - Can execute commands remotely
   - Can transfer files
   - Keeps things purely in Javascript

Cons:

- Fairly complicated to use
- Although possible, there is no well-established way to transfer files
- Third-party node.js dependency
- Must run on server instead of user's local browser
    - Prevents the server from SSHing into clusters or servers that require a VPN to access
    - Possible security risks with allowing users to SSH into arbitrary host
- Possible security risks when using user's SSH username and password

Testing:

- The ssh2 module and node.js were used to attempt an SSH connection to Monsoon, although it was unsuccessful.

Decision

We will use Python with Paramiko. Due to its simplicity, it will be easy to create, and more importantly, maintain the SSH portions of the project. If more files need to be transferred, it will be as easy as adding a few more lines of code. Additionally, the fact that Paramiko has built-in functions for both sending files and executing commands will make the it much less prone to error. Although we will need AJAX to run Python, we will already be using Python for dynamic HTML generation, so the use of Paramiko does not introduce any additional requirements. Unfortunately, it is not possible to SSH to a server via the user's local browser. As such, any servers that require a VPN to connect will not be able to be accessed (with the exception of any servers that are hosted by NAU). It may be necessary to limit the scope of the project to Monsoon (or any other NAU server) rather than an arbitrary host.

	SSH connections	**Transfer files**	**Simple**	**No third-party dependency**
Node.js	✓	X	X	X
Paramiko	✓	✓	✓	X

VISUALIZATION

Because our project is meant to make researchers' lives easier, simpler, and more efficient, it's logical that we would include a means to visualize the results of a BioNetFit run. These visualizations could be generated once BioNetFit has finished executing, and the end-user has selected from the database which files they want visualized.

1. Python via MatPlotLib module

   The third-party module MatPlotLib was created with simplicity in mind. Creating a graph can be done in just a few lines of code. It's easy to customize the types and details of each graph, allowing users to have more options when visualizing data.

   Pros:
   - Very simple and easy to use
   - Flexible
   - The use of Python keeps a unified development environment
   - Maintainable -- any new method of visualization would be easy to implement

   Cons:
   - No easy way to show/hide data
   - Third-party dependency

   Testing:
   - A program was created that could read in any BioNetFit output file and plot all variables on a single graph.

2. Javascript via D3 library

D3 is a powerful visualization engine that can be used to created a wide variety of visualizations. It gives developers an extreme degree of control over their visualizations, allowing for things like animation and uncommon data representation.

Pros:
- Fine-grained control over visualizations
- Allows for animation and unique visualization methods
- Javascript -- can be run in any browser

Cons:
- Extra control also means added complexity
- Third-party dependency
- Data gotten in Python would have to be translated into a Javascript-readable format

Decision

Python via the MatPlotLib module will be used to generate visualizations. Because the bulk of our project will be using Python, it makes sense to continue to use Python in order to keep a unified development environment and a more maintainable project as a whole. Although D3 can provide special features that MatPlotLib cannot, for the purposes of our project, these features are not needed. Showing and hiding data in MatPlotLib can be accomplished by created separate plots for each variable, and overlaying them on top of each other as transparent PNGs.

	Generate plots	**No third-party dependency**	**Animations / special features**	**No data translation**	**Simple**
MatPlotLib	✓	X	X	✓	✓
D3	✓	X	✓	X	X

LOCAL MACHINE EXECUTION

The local machine would need to execute commands via the command line to use the BioNetFit program from a local machine. However, it is a command line program and executes via the command line only. This makes it very difficult to do when using the service from a web portal. On the other hand, there could be a solution that does not use the local machine but instead a server on another machine that is used to execute the program.

1. Using Command Line directly from the web

   If possible, we could execute the BioNetFit program directly from the web on the user's local machine with a few commands and sending these commands to the BioNetFit program.

   Pros:
   - The use of the website would mean users do not have to know how to execute them, making it more user friendly
   - It would be very quick and simple for the user, so they would not need to spend time doing that themselves

   Cons:
   - Could have security issues with using command line from the web
   - If there is a security breach and the program is altered or changed for malicious content, it could be catastrophic for a system using it
   - Due to security risks, it is not possible to execute command line through the web directly
   - The user would not get as much experience using the execution from a command line

   This is not feasible, because it is not an execution of commands that is supported.

2. Using a server to execute the program

   BioNetFit needs to be executed on a local machine with a command line, it is possible to create a separate program that could be used to do this. However, it would be better design to use a server to execute the files instead.

   Pros:
   - The user does not need to install anything
   - Everything is truly web based, so information can be accessed from anywhere

- There are no system requirements that need to be met
- Slow systems can run the programs as easily as fast ones

Cons:

- This puts a load on a specific system being used
- Speed is reliant on the system used for server execution
- There could be outages that would stop all usage of the program

This is feasible, because we could set up a separate server to execute the program, and send information back to the website for users to use. This is also the only option to use, if we are to execute them at all.

Decision

We are forced to use a server execution if we are to execute the program at all. This is due to issues specifically related to running command line from the web. As it is not even possible to run command line commands from the web, it is not possible to use this method. If we are to execute the program through the web at all, then, it will have to be through a separate server. This would quite likely be a cluster server, as running such high loads of information would take long periods of time if it was not used on a cluster.

INTEGRATION

The main workload of the project consists of reading the BNGL file, dynamically generating an HTML page, using SSH to transfer the file and run BioNetFit on a cluster or server, and saving users' information in a database. These four tasks will be handled using Python. More specifically, vanilla Python will be used to parse the BNGL file, the Django will be used to generate HTML pages, Paramiko will be used to establish SSH connections, and PyMongo will be used to connect to, add, and retrieve information from the MongoDB database.

Javascript will be used minimally in the project, acting as a bridge between the website and the Python programs. For example, it will be used in conjunction with AJAX to call the Python scripts at the appropriate times. It will also be used to allow the user to upload the initial BNGL file.

All Python programs, Javascript programs, HTML pages, and databases will exist on a server located on NAU's campus. The server will need an SSL certificate in order to communicate with the users securely, especially when handling users' SSH usernames and passwords.

All modules and languages mentioned have licenses that allow for, at the very least, use in open-source projects, which our project is.

CONCLUSION

In conclusion, we have introduced our project, addressed the complications of the software design, analyzed possible solutions to the problems facing us, and explained how the selections would meld together into a cohesive unit. Each of our software decisions were carefully made, and will allow BioNetFit to become the interactive experience that our client desires. Python and Django will allow for flexible and easy development of a GUI that will vividly encompass the BioNetFit procedure. MongoDB will serve as an excellent database systemm which in addition to being easily modifiable, will work well with Python through the plugin known as PyMongo. Python will also be utilized for SSH, as it will be easily integrated to the project, as well as being a commonly used software by one of our team members. Finally, by using all of the above software, the website will run free of a local implementation. This will allow the entire process to be hosted online. For each of our solutions, we have a high level of confidence that the software will fit the problem description. We are incredibly determined that our project will be easily implemented using the above software, and any changes will be easily implemented with little confusion.