

HBaaS: Heterogeneous-accelerated Bioinformatics-as-a-Service

Mid Project Proposal, 2 December 2014

Senior Design Group 3

Stevens Institute of Technology

Sponsor: MIT Lincoln Laboratory and CSAIL

Team: Dylan Hutchison, Eric Cherin, Xin Li, Hefei Yang
"We pledge our honor that we have abided by the Stevens Honor System."

Advisor Dr. Narayan Ganesan

**Collaborators: Di Ren, Xuelian Liu, Mojie Yao,
Jaroor Modi, Peiran Guan, Hanyu Jiang**

Executive Summary

Modern applications call for solutions that handle “big data,” datasets that span multiple machines and cannot fit in main memory, and “big compute,” computation patterns that tax even the most advanced processors. Bioinformatics is no exception.

We present HBaaS, Heterogeneous-accelerated Bioinformatics-as-a-Service. Our platform leverages heterogeneous computer architectures to provide sequence matching and motif finding as a web service for users of bioinformatics data. We tackle big data by means of the Accumulo distributed database, and big compute by means of GPUs. Their integration delivers top-tier performance for our chosen applications.

Table of Contents

Section – I: Project Definition and Plan (initial in Fall, refined in Spring)

I.1 Mission Statement

I.2 Background

I.3 Stakeholder List

I.5 Project Scope and Resources

I.5a Budget

I.6 Project Plan

Section – II: Design, Evaluation & Optimization (initial in Fall, refined in Spring)

II.1 Requirements

II.2 Constraints and Assumptions

II.3 Applicable Codes/Standards/Regulations

II.4 Concept Development and Selection

II.4a Website and Web Server Selection

II.5 Preliminary (Fall) and Detailed Design (Spring)

II.5a High-level HBaaS pipeline

II.5b Data Sources

II.5c Accumulo Table Design

II.5d Computing alignments with HMMER

II.5e Aligning and Scoring scanned sequences

II.5f Alternative Designs

II.6 Design Evaluation Methods

II.7 Design Evaluation Report: Performance, Reducibility and Cost

II.8 Design Revisions and Optimizations (Fall/Spring)

[II.9 Final Design Specification with BOM \(spring\)](#)

[Section – III: Entrepreneurship & Business Development \(primarily TG course requirements\)](#)

[III.1 Business Objectives and Risks \(Fall\)](#)

[III.2 Competitive Intelligence: Market Analysis \(Fall\)](#)

[III.3 Lean Canvas Business Model \(Fall\)](#)

[III.4 Financial Analysis \(Spring\)](#)

[III.5 Intellectual Property \(Spring\)](#)

[Section – IV: Results](#)

[IV.1 Conclusions](#)

[IV.2 Recommendations](#)

[Appendices](#)

[A Team organization chart](#)

[B Project Gantt Chart](#)

[D Design Documents: Drawings, Layouts, Analysis reports](#)

[E Team Logistics Systems](#)

[F References cited](#)

[Conference and Journals of Interest for publishing results](#)

[G Example Data](#)

[gbpri1.fsa_aa -- FASTA file from NCBI GENBANK containing protein sequences](#)

[globins4.hmm -- example motif Hidden Markov Model file](#)

[globins4.out -- Exmple scores for sequences against a motif HMM, followed by alignments](#)

Section – I: Project Definition and Plan (initial in Fall, refined in Spring)

I.1 Mission Statement

We aim to create an online Bioinformatics-as-a-Service (BaaS) platform, a web service providing sequence and motif alignment and search for proteins, accelerated by GPUs and the Accumulo DB. Our platform will run on a cluster of machines equipped with GPUs. Each machine will host data as part of the Accumulo distributed database and leverage their attached GPUs to accelerate parallel computation and data retrieval.

I.2 Background

Bioinformatics-as-a-Service (BaaS)

Bioinformatics is the use of mathematics and computer science to organize, analyze, and store the data generated in life science research and by the health industry. Bioinformatics often involves the development of software tools to understand biological processes through applications such as data mining, sequence analysis, gene and protein expression, protein structure modeling, and network and systems biology. Examples of bioinformatics applications include RNA and DNA sequencing, genome alignments and assemblies, mutation identification, microarrays, and database creation and management.

The very large amount of data that is collected in biological studies such as genomics or protein analysis must be processed using computers. Computer scientists combine algorithms, statistics, and mathematics, and engineering in order to make sense of the information.

HMMER is used for searching sequence databases for homologs of protein sequences, and for making protein sequence alignments. It implements methods using probabilistic models called profile hidden Markov models (HMMs).

Compared to BLAST, FASTA, and other sequence alignment and database search tools based on older scoring methodology, HMMER aims to be significantly *more* accurate and *more* able to detect remote homologs because of the strength of its underlying mathematical models [4, 5]. In the past, this strength came at significant computational expense, but in the new HMMER3 project, HMMER is now essentially as fast as BLAST.

We call the HMMER algorithm on data queried from Accumulo [6] using local graphical processing units (GPUs).

GPU

A graphics processing unit (GPU) is designed to rapidly manipulate and alter memory in order to accomplish a particular task. GPUs are commonly used in devices that require image processing such as computers, smartphones, and work stations. GPUs have a highly parallel structure which makes them very effective for processing data that can be done in parallel. Researchers such as Burrage et al sped up biological computation with GPUs [7] .

We use GPUs for accelerated computation on data queried from accumulo. They are ideal because the HMMER algorithm is SIMD-- single instruction multiple data-- meaning that the same instructions can execute in parallel across all the data on a GPU.

Apache Accumulo

The Apache Accumulo sorted, distributed key/value store is a robust, scalable, high performance data storage and retrieval system. Apache Accumulo is based on Google's BigTable design and is built on top of Apache Hadoop, Zookeeper, and Thrift. Apache Accumulo features a few novel improvements on the BigTable design in the form of cell-based access control and a server-side programming mechanism that can modify key/value pairs at various points in the data management process.

In short, Accumulo is our database, delivering fault tolerance, distribution and availability for huge amounts of protein sequence and model data. Cell-based security may gain relevance if we need to restrict access to certain gene data.

I.3 Stakeholder List

Our target users range among physicians working in personalized medicine, forensic scientists working in crime scene identification, epidemiologists working in disease identification, genealogy consultants working in kinship analysis, biologists working in research generally, and other groups who use protein model-to-sequence scoring in their daily work. These users need scoring information fast, so that they may prescribe correct medication and catch criminals early, provide expedited genealogy services and more rapidly discover scientific advances.

Additional stakeholders are other bioinformatics-as-a-service providers. Janelia, a research group at the Howard Hughes Medical Institute (<http://www.hhmi.org/>), is the closest provider, hosting the online framework <http://hmmmer.janelia.org/>. The Janelia group would be interested in our project, particularly our design, if we achieve sufficiently greater performance. The NCBI would also be interested if our performance increase is much greater. The NCBI hosts a similar protein sequence tool called BLASTp (<http://www.ncbi.nlm.nih.gov/blast/Blast.cgi?PAGE=Proteins&PROGRAM=blastp>), which has the same function as HMMER but uses a different algorithm.

I.4 Project Scope and Resources

We will provide the end user with a website that can be used to help deal with protein sequences . We will update the website based on customers needs and add new features such as additional hmmer classes. We will add more gpu nodes if it is necessary to expand.

We aim to provide two service classes: *hmmsearch* and *hmmbuild*.

- In *hmmsearch*, a user provides HMMs (Hidden Markov Models) representing protein sequence motifs, and requests the top-scoring sequences for each model from existing, curated databases of protein sequences stored on our Accumulo platform. Users may alternatively store motifs in a server-side database.
- In *hmmbuild*, a user specifies a range of sequences and requests the HMM motif that best represents the consensus between the sequences. The resulting motif may be added to a motif database.

hmmbuild is a reach goal; if we run out of time we will not go beyond *hmmsearch*.

All our software resources are open source. In terms of hardware resources, we have three levels:

- A server located in Stevens Burchard room 412, equipped with three GPUs. This is known as *b412srv*.
- A research cluster of 8 machines, all equipped with GPUs, located in the Stevens library basement.
- Access to very large clusters at MIT CSAIL

I.4a Budget

We have no budget as we have no material expenses. Hardware is supplied by Stevens and MIT CSAIL and software is open source. Perhaps we will buy some poster board. All our costs are in human development time.

I.5 Project Plan

Here is our plan, broken into major goals with subgoals:

1. Web server frontend
 - a. Create a basic set of web pages with project information
 - b. Create a test webpage for use in development. The test webpage will display the results of internal queries to make sure they work.
 - c. Create a production-level query webpage containing a form to enter in sequences, motifs and target databases.
 - d. Create a file upload mechanism as an alternative to entering text.
 - e. Create javascript to validate entered information.
2. Web server backend

- a. Setup our backend on our compute server b412srv or open required ports on b412srv to access it remotely. Determine which is the better strategy.
 - b. Test that the backend receives information correctly from the front end.
 - c. Determine a way to return results to the frontend asynchronously, since some query jobs will take a long time (on the order of hours).
 - d. Setup the server for multithreading, to handle multiple jobs concurrently.
 - e. Write code to connect to Accumulo and scan the tables
 - f. Integrate in the custom iterators described below.
3. Accumulo setup
 - a. Do prerequisites; create user accounts, install Hadoop and Zookeeper
 - b. Install Accumulo on our server b412srv on our 1TB networked filesystem.
 - c. Design an Accumulo schema to store protein data.
 - d. Create a client to parse raw sequence data and ingest the sequences into Accumulo
 - e. Create a client to parse taxonomy information and ingest the data into Accumulo
 4. Accumulo Iterators
 - a. Create test iterators to ensure everything is working
 - b. Learn how to run the HMMER C code in development at Stevens
 - c. Create an iterator that uses JNI to call the HMMER C code, single-core CPU version
 - d. Create an iterator that uses JNI to call the HMMER C code, multi-core CPU version
 - e. Create an iterator that uses JNI to call the HMMER C code, single GPU version
 - f. Create an iterator that uses JNI to call the HMMER C code, multi GPU version
 5. Integration
 - a. Unit test each of the above components
 - b. Complete the pipeline-- data flow from a request at the front end to the back end to Accumulo scanning to Accumulo iterators to the back end to the front end
 - c. Benchmark everything. Compare to Janelia's HMMER service specifically: <http://hmmmer.janelia.org/>
 - d. Work on the slowest components from benchmarking. If performance is ok, add features and datasets.
 - e. Write up results.

Section – II: Design, Evaluation & Optimization (initial in Fall, refined in Spring)

II.1 Requirements

HBaaS is a web service to answer protein sequencing queries. For example, given a user-provided HMM motif, what are the top 10 protein sequences in the NCBI Genbank

database [3] that match that motif? What are their alignments? What if we restrict the database to only proteins from the species homo sapiens?

To answer these kinds of queries,

- HBaaS must have a web form where users can enter and submit queries
- HBaaS must be able to process queries from multiple users in parallel (typically solved by a multithreaded web server)
- HBaaS must be able to return results to the user asynchronously, since some queries can take several hours to complete.
- HBaaS must be able to accept new data (protein sequences or HMM motifs) to ingest into the web server's backend database.

II.2 Constraints and Assumptions

The web service will not process more requests than we can handle with our cluster resources. We will know this constraint better once we start benchmarking.

We will not store more data than we have hard disk space, about 1TB on Dr. Ganesan's server.

We assume data and queries are in correct format. We don't want to error check every sequence, model and query as this costs development time and possibly performance.

We assume the result size of a query is relatively small, on the order of a handful of sequences or motifs that are the best matches for a given query. We are not in the business of transferring entire databases over the Internet.

II.3 Applicable Codes/Standards/Regulations

OpenCL (<https://www.khronos.org/opencv/>) is an API industry standard for GPU computation. CUDA is another API specific to NVIDIA. We aspire to conform to OpenCL to guard against vendor lock-in but will use CUDA at first since it has greater support (more libraries, more examples, more of our team has background knowledge in CUDA).

FASTA (<http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml>) is an industry standard file format used for protein sequences and HMM motifs. We should be capable of handling input data written in the FASTA file format.

II.4 Concept Development and Selection

Our original project idea was to create a distributed graph library, offering graph algorithms on the Accumulo database that leverage local GPUs. GPUs accelerate computation and Accumulo enables scalability to big datasets. After investigating the graph

library idea, we decided that it was too abstract.; the team had trouble connecting with the project idea and motivation.

This sparked our shift to specialize on an application in bioinformatics. Dr. Ganesan and other students had some experience with biological computation, and everyone found the application more inspiring. Our central concept remains the same-- that we will use **GPUs to accelerate computation** and **Accumulo to store and operate on massive data sets**.

Dr. Ganesan and Hanyu, as part of the [Heterogeneous Computing and Mathematical Modeling](#) group, have been developing HMMER for some time. HMMER is a C program that solves hmmsearch by doing sequence match scoring computations. Their NVIDIA GPU version of HMMER succeeds on parallelizing the search using multiple GPUs. We chose our architecture such that we could build on the existing HMMER work and use it. No need to develop solutions to protein sequence-motif scoring from scratch.

We decided to use a database in our architecture because user needs call for selective queries. It is helpful to users to do scoring over only the proteins within a selected group. For example, a user may wish to find the best motif for all the plant protein sequences or all the homo sapiens protein sequences. Databases give the ability to scan over targeted parts of the data, as opposed to non-database solutions that generally map over an entire dataset (e.g. MapReduce). How we are able to select certain subsets of data in a dataset depends on the database table format we use for storing data. Our table format is detailed below.

We chose Accumulo as our database because some team members had background knowledge of it and it offers iterators, a framework to do computation on data as it is scanned from a database. The beauty of iterators is that they execute in parallel across all tablet servers that retrieved data is stored at. We saw a natural opportunity to call HMMER code (which uses GPUs) from iterators.

With GPUs and Accumulo at the center of our design, the remaining component to link everything together is our web server.

II.4a Website and Web Server Selection

Our web platform process starts with the web front-end, so it is necessary to have a good front-end design and development. For the front-end, we will first have three most important features which are Data Upload, Result Download, and Query Form. Data Upload function will be used by us, the designer; user would frequently use the result download and query form functions. Basically, users would input in the query form to start the process, and they will get the results through the web server technology, which is Java and Apache in our case. There are some candidates for the web server, which are Apache, IIS, C/C++, Ruby, Node.js, and Java.

Apache is at the core of the LAMP technology stack upon which a lot of server architecture is based: Linux, Apache, MySQL, and PHP/Python/Perl. It is a traditional web

server that is very easy to configure but may require plug-ins or backend scripts, like ASP and PHP, when complex functions are needed.

On the list of IIS advantages, its Active Server Pages(ASP) percolates to the top. ASP enables developers to embed code into HTML pages. These ASP pages are parsed by the server before being supplied to the client as HTML. ASP enables developers to work in a number of different .NET languages, including C/C++, and C#. IIS is a good alternative for Apache but is far more expensive once we add in the cost of Microsoft Windows Server 2008 R2 and other back engine software.

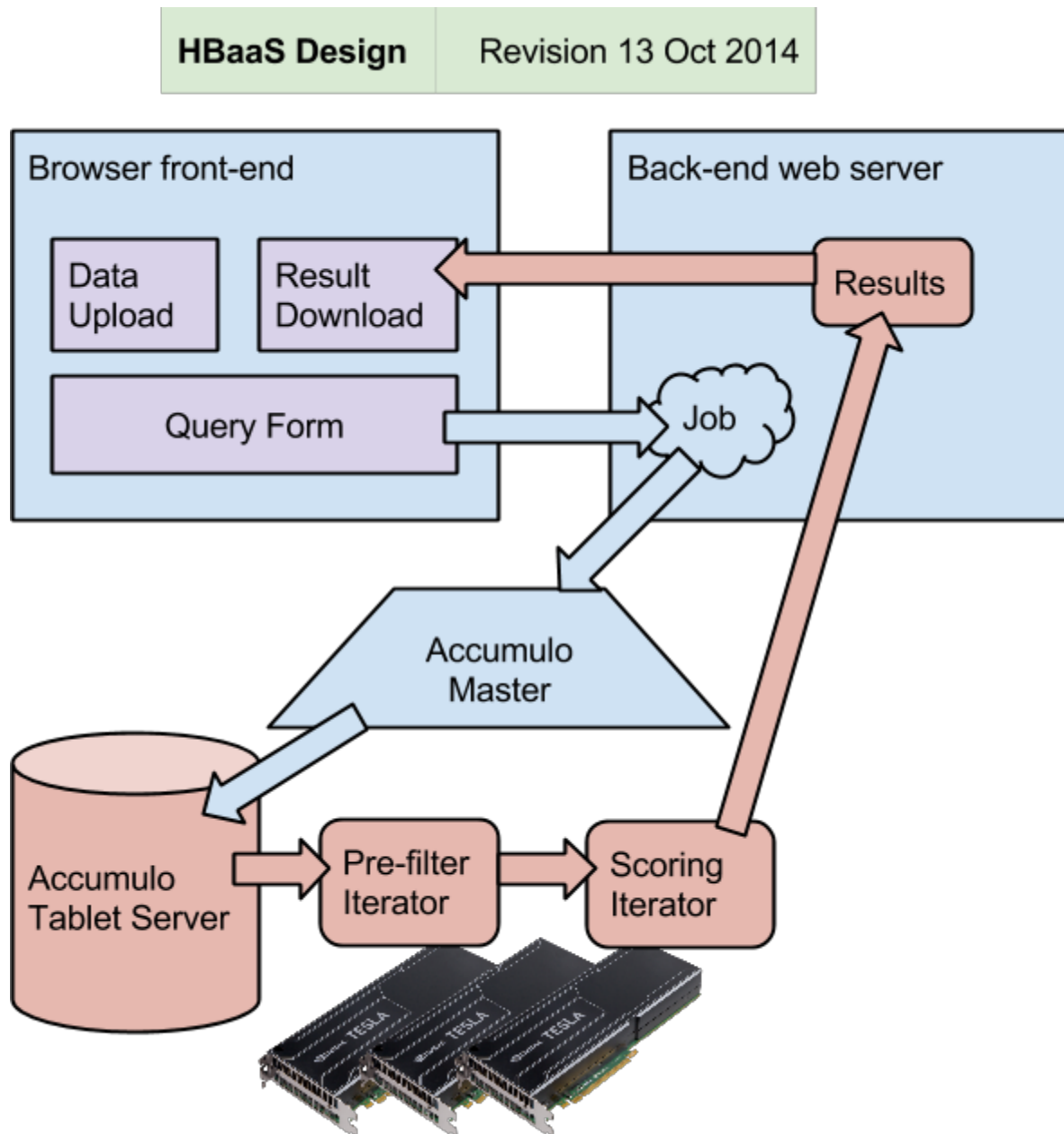
C/C++ is powerful but it is so underlying that requires a huge amount of details and specifications. C/C++ does not have existing libraries for constructing web server and offers manual memory management.

Ruby and Nodejs are both powerful software offering automatic memory management and a variety of libraries for constructing web server, which could save the group more time on functionalizing the web server.

Besides the advantages Ruby and node.js have, Java offers certain functions such as readymade API that make it convenient to exchange data with server backend, which is Accumulo Database and GPUs in our case.

Also, even choosing web server language, it would eventually go through Java platform. Therefore, considering the time and money that Apache could save and convenient function that Java could provide, the group has been applying Apache and Java for web server construction by now.

II.5 Preliminary (Fall) and Detailed Design (Spring)



II.5a High-level HBaaS pipeline

1. A user submits a query via our web interface.
2. Our web server backend creates a customized query-and-compute request for the Accumulo database and launches it.

3. In parallel, each machine on the Accumulo database searches through its locally stored protein sequences. The top scoring results are returned to the web server backend.
4. The web server backend forwards the results to display on the web interface.

II.5b Data Sources

Our initial protein database is NCBI's Genbank database, available for public download at ftp://ftp.ncbi.nih.gov/ncbi-asn1/protein_fasta.

Proteins are most commonly identified by an accession number (also referred to as a seqID). Some organizations like the NCBI also have their own identifier for cases when an original sequence uploader re-uploads the same sequence due to an error. Additionally, organizations that aggregate proteins from multiple databases identify the original source database. We use the software BioJava3 (<http://biojava.org/>) to parse FASTA protein sequence files, as it saves us from writing our own parser [8].

Taxonomic information about the protein sequences is also taken from NCBI at <ftp://ftp.ncbi.nih.gov/pub/taxonomy/>. This includes data linking sequences to the species identifier that that sequence is from, data on the standard biological hierarchy of identifiers from kingdom down to species and other related information. An example taxonomic lineage for humans is shown to the right.

Sample Abridged Taxonomic Hierarchy
1 Eukayota
2 Metazoa
3 Chordata
4 Craniata
5 Verebrata
6 Mammalia
7 Primates
8 Haplorrhini
9 Hominidae
10 Homo
11 Sapiens

II.5bi Data Ingest Note

Most of the data ingest is straightforward: reading the data from the NCBI and other institutions' files in their correct data formats, and ingesting the parsed data into the database in the table schema described below.

There is an additional step required for linking protein sequences to their taxonomic identifiers. The NCBI lists sequence's taxIDs by the genbank identifier 'gi.' We wish to store the taxonomic listing by accession ID, because the accession ID is a universal identifier for a protein sequence. We therefore look up in the Accumulo database accession IDs that corresponds to genbank identifiers. This links protein sequence accession IDs to taxonomic IDs.

II.5c Accumulo Table Design

The format we store protein sequence data in an Accumulo table determines how easy it is to answer some queries over others. This is called table design for query planning. For example, if we knew that a very common query is "What are all the sequences associated with a certain family," we could create an index structure to make the data access time for that query optimal. The tradeoff is either redundant data storage or slower access to other queries. Benchmarking will help indicate whether database access patterns are a performance bottleneck.

We chose our table design based on the following:

- There is far less data, in terms of bytes, in the sequence identifier and taxonomic information than the actual protein sequence. Therefore we should avoid repeating sequence data, whereas it is okay to repeat identifier and taxonomic data.
- We expect users to request alignment computations on all the sequences for a given taxonomic level. For example, what are the best scoring sequences within all Mammalia class sequences in the database? It should be relatively straightforward to answer this query.
- It should be easy to answer the reverse question: what is the taxonomic hierarchy for a given sequence? This is a common question after finding the best matching sequences, to discover what groups those sequences are from. Are they all viral sequences, for example?
- We should provide a way for users to search for a taxonomic ID by name, since we do not expect users to memorize that 9606 is the taxID for homo sapiens, for example.
- Accession number is the most common identifier for a protein sequence, though searching by genbank ID and by source database are good features to have.

To facilitate answers to the above expected questions, we base our schema on a design pattern called the D4M Schema [11]. Specifically, we use the pattern of storing a table alongside its transpose for the table containing sequence identifiers and taxonomic information. This double data storage, which is acceptable because the amount of data there is far smaller than the sequence data itself. In return, we gain full indexing across all rows and columns. We show how to answer the two questions below.

TableName	Col1	Col2	...
Row1	Val11	Val12	...
Row2	Val21	Val22	...
...

Here is our table design, written in the format:

		child							parent	
Ttax		sciname root	sciname Bacteria	othername all synonym	othername Prokaryota in-part	rank no rank	rank superkingdom	taxid 1	division Unassigned	division Bacteria
taxid 1	1	1	1	1	1	1	1	1	1	1
taxid 2			1		1		1	1		1
accid								1		

TtaxTDeg	degChildTax	degChildAcc
taxid 0000001	1	1

TtaxDeg	deg
taxid	2
accid	1

parent		child	
TtaxT		taxid 1	taxid 2
sciname root	1		
sciname Bacteria			1
othername all synonym	1		
othername Prokaryota in-part			1
rank no rank	1		
rank superkingdom			1
taxid 1		1	1
division Unassigned	1		
division Bacteria			1

TseqRaw	seq	desc
accid BAC05839.1	MAAXA...	EGF rep...
accid AAB63305.1	CLNNF...	coded f...
accid AAB54048.1	CFRCG...	partial...
accid AAB67604.1	TVKLLL...	seven t...
accid AAB71326.1	LGAGES...	Rod tra...
accid AAB71327.1	TIEEGT...	Sonic H...

Tseq	gi 021928500	gi 001809231	gi 001809235	gi 001613900	gi 002341018	gi 002439516	db gb BAC05839.1	db gb AAB63305.1	db dbj AAB54048.1	db gb AAB67604.1	db gb AAB71326.1	db gb AAB71327.1
accid BAC05839.1	1						1					
accid AAB63305.1		1						1				
accid AAB54048.1			1						1			
accid AAB67604.1				1						1		
accid AAB71326.1					1						1	
accid AAB71327.1						1						1

TseqT	accid BAC05839.1	accid AAB63305.1	accid AAB54048.1	accid AAB67604.1	accid AAB71326.1	accid AAB71327.1
gi 021928500	1					
gi 001809231		1				
gi 001809235			1			
gi 001613900				1		
gi 002341018					1	
gi 002439516						1
db gb BAC05839.1	1					
db gb AAB63305.1		1				
db dbj AAB54048.1			1			
db gb AAB67604.1				1		
db gb AAB71326.1					1	
db gb AAB71327.1						1

TseqTDeg	
gi	6
db gb	5
db dbj	1

Here's a few notes on interpreting the tables:

- The suffix 'T' in TseqT and TtaxT stands for *transpose*.
- taxid fields are zero-padded to 7 characters (not shown in table)
- TtaxDeg contains a count of the number of taxids in the database and the number of accIDs linked to taxIDs.
- TtaxTDeg contains counts of how many taxonomic identifiers and how many accession IDs are childs of each taxid. A missing value means 0.
- 'sciname|Bacteria' denotes the unique scientific name for a taxonomic entry.
- 'othername|Prokaryota|in-part' denotes that Prokaryota is another (non-primary) attribute for that taxonomic entry of type 'in-part.' Other types are synonyms, authorities, common names and others.
 - Including the attributes for a taxonomic entry allows a user to search the taxonomic database for a taxonomic level by a perhaps more easily recognizable name, such as finding the taxID for the scientific name *Canis lupus familiaris* by the common name *dog*.
- 'accid|BAC05839.1' is an accession number.
- 'gi|21928500' is an NCBI identifier, assigned sequentially from the order that sequences are processed by the NCBI. [source] We zero-pad the 'gi' field to 9 digits in order to group numbers like '10001' before '100000001'
- 'gb' refers to the NCBI Genbank database. 'dbj' refers to the DNA Data Bank of Japan (<http://www.ddbj.nig.ac.jp/>).

- Table TseqTDeg contains a list of all the databases present and a count of how many sequences are in each database
- A '1' value indicates the presence of a column for a row. This is used when the column name contains the information traditionally stored in the value. Storing the data in the column instead of the value allows the database to index that value for significant search speedup.
- The column 'seq' contains raw sequence data, without any further processing.
- The column 'desc' contains the header description for a sequence. These headers are an ad hoc description of a sequence and do not have a standard format. Therefore, we do not index them for searching.

Example queries:

- Find all the sequences in the database under a given taxID
 - Scan TtaxT repeatedly to get child taxids.
 - TtaxT also lists the accid sequences under their taxid.
- Build the taxonomic hierarchy of sequence 'accid|AAB71326.1'.
 - First scan Ttax on row 'accid|AAB71326.1' for the column starting with 'taxid|'. The result is the taxid for that sequence.
 - Scan Ttax repeatedly to get the parents of each next taxonomic level.
- Find the taxonomic IDs associated with names beginning with 'homo'. (prefix search)
 - Scan TtaxT on rows beginning with 'sciname|homo' and optionally 'othername|homo'. Result are the taxonomic IDs and any accession numbers for sequences directly under that taxid.
 - The other names for 'homo' may be found by scanning Ttax with the newly found taxid.
- Find all sequences from the DNA Data Bank of Japan.
 - Scan TseqT on rows beginning with 'dbjj|' for all columns. Result are the accession numbers for all relevant sequences.
 - Scan TseqRaw on the row accession numbers from (a) for column 'seq' to obtain the actual sequences.

Possible points of expansion that we will consider if we have time and use case demand:

- We may add extra degree columns or degree tables to index the total count of sequences under a given taxonomic ID. Currently this is possible by scanning through the data and counting by recursive algorithm. For example, we would to count the number of sequences for a genus, we would find all species childs of that genus (table scan pass 1) and then count the child sequences of those species (table scan pass 2). Indexing the data reduces the access time to a constant.
- We may index the sequence header description by word, in order to find the sequences with a header that contains the words "helix receptor" for instance.

- Our design is flexible in that it allows the addition of properties associated with a taxID or a seqID, in case we find indexing such properties useful in the future. For example, we may store 'sampleDate' or 'seqQuality' as additional columns in Tseq.

II.5d Computing alignments with HMMER

HMMER is a program written in C to compute sequence alignments and scores. In order to call HMMER from Java code, we use a technology called JNI, Java Native Interface. JNI allows a Java method to call a C method, passing parameters and return values in memory.

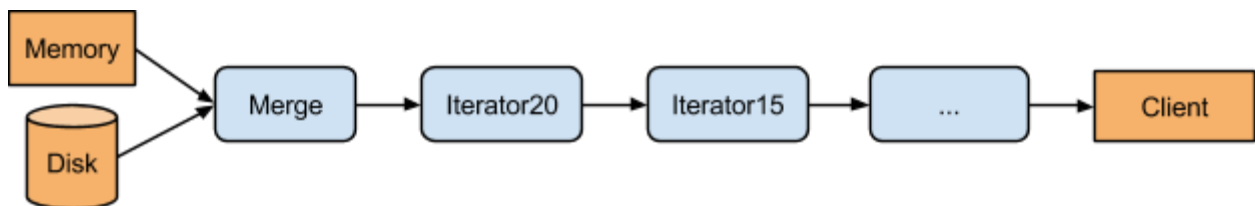
HMMER comes in several versions. The first original version runs in parallel on a CPU. We use the CPU version for our first prototype because it is the simplest. A version of HMMER in development at Stevens' Heterogeneous Computing and Mathematical Modeling group runs on GPUs. This is our final target, to leverage GPUs for computational acceleration as part of an Accumulo platform that handles mass data.

II.5e Aligning and Scoring scanned sequences

While the Accumulo schema works well to select only sequences of interest, our goal is to compute sequence alignments, not to find sequences. Our use case is to find the best scoring sequence alignments for a given set of queried sequences.

One way to proceed is to gather all the queried sequences onto a single machine at the server and perform the computation on that machine. We will implement this approach for comparison purposes but it is not our final goal, because it ignores potential distributed computation of sequence alignments and unnecessarily transmits all the sequences over a local network. The only necessary transmissions ought to be the highest scoring sequences.

Our target design will compute sequence alignments locally, at the machine the data is stored at. We will do this by using Accumulo's iterator framework. Here is a diagram illustrating how scan-time iterators work in Accumulo:



The arrows here represent streams of data. The blue objects are in-memory processes and the orange objects are final data destinations.

Data originally resides in disk or in memory (if cached). All data matching the criteria of a scan is streamed and merged together in sorted order through an internal merging iterator. From there, data passes through a hierarchy of iterators, some system-defined and

some user-defined. Iterator 20 is the typically a versioning iterator that filters away old data. We define a custom iterator after that at, say, priority 15 (higher priority iterators execute first).

Our custom iterator operates on data in three phases:

1. Prefiltering. A quick computation that filters out sequences with very low alignment scores.
2. Scoring. An intensive computation that returns a score (and an alignment) given a sequence.
3. Maxing. Only lets pass the top X scoring alignments, where X is query-defined (say 10).

For increased efficiency, we define our iterator to operate in batches. It will handle some large number of scanned sequences (TBD, for example 5000) at a time. This allows the HMMER platform to take advantage of parallelism by running on multiple sequences at once (SIMD style-- Single Instruction Multiple Data).

Keep in mind we have an outer level of parallelism since iterators simultaneously run on all tablet servers that store queries data. In order to maximize this parallelism, it is important that the protein sequence data is evenly distributed across all available tablet servers. We will gauge our performance by measuring load balance when we perform benchmarking. If necessary, we have three options to introduce load balance:

- Table splits. We designate places within the database to explicitly divide data onto different tablet servers. Accumulo usually does a good job determining tablet splits on its own (except during ingest phase), but we may manually set these if we identify a common query pattern does not load balance.
- Reversing sequence IDs. Because Accumulo stores rows in lexicographic order, reversing the sequence IDs will better shuffle rows around tablet servers. For example, the rows 'abx' and 'acb' are adjacent. When reversed, the rows 'xba' and 'bca' are on opposite ends of the alphabet and almost surely on different tablet servers.
- Using a hash function. If necessary, hashing seqIDs uniformly throughout the alphabet will guarantee load balancing, at the price of an additional computation.

II.5f Alternative Designs

One alternative to the Accumulo database is no database: just store our protein data inside a big data file system like Hadoop. We can then use a framework like MapReduce to run computations across all the data in Hadoop. MapReduce is a better design if our computations really do run on the entire database [10]. This is not our use case; we hope users will take advantage of selective access to data by taxonomic classification to only run computation on a subset of data. We may be able to offer the best of both worlds by running

MapReduce jobs from Accumulo, for the cases where we need to score alignments on a whole database.

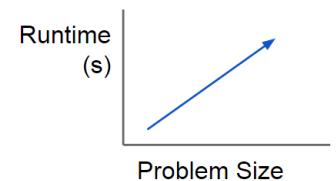
Another alternative is using a SQL database such as BioSQL [9]. SQL databases differ from NoSQL databases like Accumulo in that SQL databases provide more built-in services in exchange for lower performance. For example, the ability to JOIN tables of a database is built into SQL database engines whereas NoSQL database designers typically need to write routines to perform a JOIN by hand. Another way to think of the difference is that NoSQL databases provide finer control over storage format and access patterns yielding better performance, at the expense of additional development time. We choose a NoSQL database since performance is our primary concern.

II.6 Design Evaluation Methods

Benchmarking is the name of the game. Since our project is an online bioinformatics as a service platform, which works like a web search engine, runtime of the program is the most important criteria that can help us determine if our design is good or not. We will measure performance in terms of runtime and plot scalability of run times against query size, motif HMM size, and number of computing resources available.

- Runtime vs Query Size and Motif HMM Size by Weak Scaling

If users send a large number of protein sequences to our web server to make a request, the back end will spend a lot more time on computing. We therefore need to find out how runtime increases with the size of problem we process.



Weak scaling is defined as how the run time varies with the number of processors for a fixed problem size *per processor*. We test weak scaling by holding problem size (query size and motif HMM size) assigned to each processing element (GPU on a database tablet server) constant and adding additional elements to solve a larger total problem. This type of measurement is a good indicator for performance because it tests how well we can meet additional demand by adding more hardware resources.

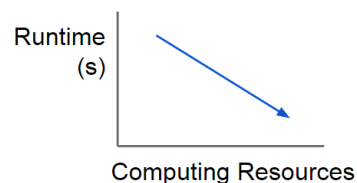
Ideal: Linear Growth
Actual: TBD

- Runtime vs Computing Resources by Strong Scaling

We use parallel processing in our design. Parallel processing is the simultaneous use of more than one GPU core to execute a program, which splits up a programming task into sections. Ideally, parallel processing makes programs run faster because there are more GPUs running it, and thus runtime will decrease linearly as we add more nodes. However, in practice, it's often difficult to divide a program in such a way that separate GPUs can execute different portions without interfering with each other. As more processing nodes are added, each node will spend more time doing communication than useful processing. At some point,

parallel slowdown occurs and adding more nodes will not help significantly. In other words, the improvement will no longer be linear.

In this benchmark we hold problem size (query size and motif HMM size) fixed and increase number of processing elements (GPUs and tablet servers). The goal is to find a "sweet spot" that allows the computation to complete in a reasonable amount of time, yet does not waste too many cycles due to parallel overhead. We aim to achieve as close to *strong scaling* as possible, when our performance (in terms of work units completed per unit time) scales linearly with number of resources used.



Ideal: Linear Speedup
Actual: TBD

- Artificial and Real World Benchmarks

We will use artificial workloads to stress test and benchmark performance to see if the system meets our expectations. We will apply a standard benchmark currently applied by other Baas platforms, such as BioBench [14]. BioBench is a benchmark suite for a variety of bioinformatics applications, including BLASTn, BLASTp and most relevantly, HMMER. BioBench uses HMMER v2.3 to search the SwissPROT protein database against the consensus of a small selection of protein sequences. Because our platform is based on HMMER, we will use a similar set of tests to BioBench as a standard to benchmark our platform before use.

After successfully running results on BioBench, we will move to real world benchmarks. At this stage, we will open our web site and put our product into use. By inviting volunteers to use our service and monitoring performance of our program, we will locate bottlenecks and make targeted improvements.

II.7 Design Evaluation Report: Performance, Reducibility and Cost

We will compare against HMMER's web service at <http://hmmer.janelia.org/search/hmmsearch> as a baseline. We expect performance improvements as a result of using Accumulo and GPUs.

II.8 Design Revisions and Optimizations (Fall/Spring)

Our initial Accumulo schema did not include nearly as much protein metadata. We had no taxonomic information or information on the source database of a sequence, for example. We added this additional information because (1) a database only adds value when used for indexing and (2) restricting queries to only sequences from a certain database or a certain taxonomic subgroup is useful to end users.

Our initial plan also included storing motifs in the Accumulo database. We decided to deprioritize motif storage and require the user to upload their own motifs (which are of far smaller size than sequences). We will revisit motif storage time permitting.

II.9 Final Design Specification with BOM (spring)

Section – III: Entrepreneurship & Business Development (primarily TG course requirements)

III.1 Business Objectives and Risks (Fall)

- **Business Objectives:**
 - Build the **fastest bioinformatics-as-a-service platform**
 - Promote heterogeneous solutions for speeding up real world processes
- **Risks:**
 - Development time higher than expected
 - Electricity bills for running servers too high
 - **We don't achieve the performance we expect.** In this scenario, our contribution is a detailed analysis of why we don't achieve good performance. We have reduced our risk by researching database and GPU architectures. Our expectations have a solid theoretical basis.

III.2 Competitive Intelligence: Market Analysis (Fall)

Other companies offering data analytics on Accumulo are Sqrrl and Argyle data [1, 2]. Their products are Sqrrl Enterprise and ArgyleDB, respectively. Companies like these two create general analytics platforms and offer some solutions to specific problems. Argyle's prime customers are financial companies in the context of fraud detection. Sqrrl's prime customers are in the healthcare industry, and other industries that need a strong privacy / access control layer on their data.

We target a very different customer base, providing a platform for specific problems in bioinformatics. We also distinguish ourselves from them by using GPUs, which few have used alongside databases. This is particularly relevant for problems with computational bottlenecks, not just data bottlenecks. (big compute in addition to big data)

Two competing tools for protein sequence computations are HMMER and BLAST, affiliated with the Janelia group at the Howard Hughes Medical Institute and the NCBI, respectively. We aim to implement HMMER on GPUs within a database, so we expect our performance to scale higher than vanilla HMMER. We also expect our web server performance to improve over Janelia's compute farm by using the modern Accumulo database to index data. We're not sure where we will stand with respect to BLAST as their BLAST uses a different scoring algorithm.

III.3 Lean Canvas Business Model (Fall)

- **Problem:**
 - Big data: the volume of protein sequence data is huge
 - Big compute: computing alignments is computationally expensive
 - Need fast and accurate protein sequence from searching
- **Solution:**
 - Leverage the Accumulo database to tackle big data
 - Use GPUs to tackle big compute

- Visit our website to get a wonderful protein searching experience
- **Customer Segments:**
 - Biologists -- for research
 - Forensic scientists -- crime scene identification -- catch criminals early
 - Epidemiologists -- disease identification -- stop epidemics early
 - Physicians -- prescribe correct, personalized medication
 - Genealogy consultants -- expedited kinship analysts services
 - Anyone using *protein model-to-sequence scoring* in their daily work
- **Unique Value Proposition:** We aim to create an online BaaS platform with improvement: a web service providing protein sequence-to-motif alignment, leveraging the *Accumulo database to tackle big data* and *GPUs to tackle big compute*.
- **Channels:** Our website and search engine, publish performance numbers in industry journals, directly market to companies, and use word of mouth in our specialized community.
- **Cost Structure:** Pay salaries and purchase hardware.
- **Revenue Streams:** Since we aim to open-source our program, primary revenue is by companies paying for a support agreement for us to help them setup, use, and maintain our program. Revenue also comes from advertisements we will have on our website.
- **Key metrics:** We will measure performance in terms of runtime and plot the scalability of run times against query size, motif HMM size, and number of computing resources available.
- **Unfair Advantage:** No one in the competition has combined the Accumulo database with GPUs as we will, and as such we expect unrivaled performance.

III.4 Financial Analysis (Spring)

We have no budget; see project scope and resources in Section 1.

III.5 Intellectual Property (Spring)

We do not intend to protect intellectual property. Everything we create is open source.

Instead, the project will derive profit by selling a service: a support / maintenance contract for a specified time period (say 1 year). This is the same business model that Red Hat (<http://www.redhat.com/en>) uses for selling support to Linux and that Hortonworks (<http://hortonworks.com/>) uses for Hadoop.

We hope many others “steal our idea” and build off it, deriving better products and citing our work, following academic and software development common courtesy.

Section – IV: Results

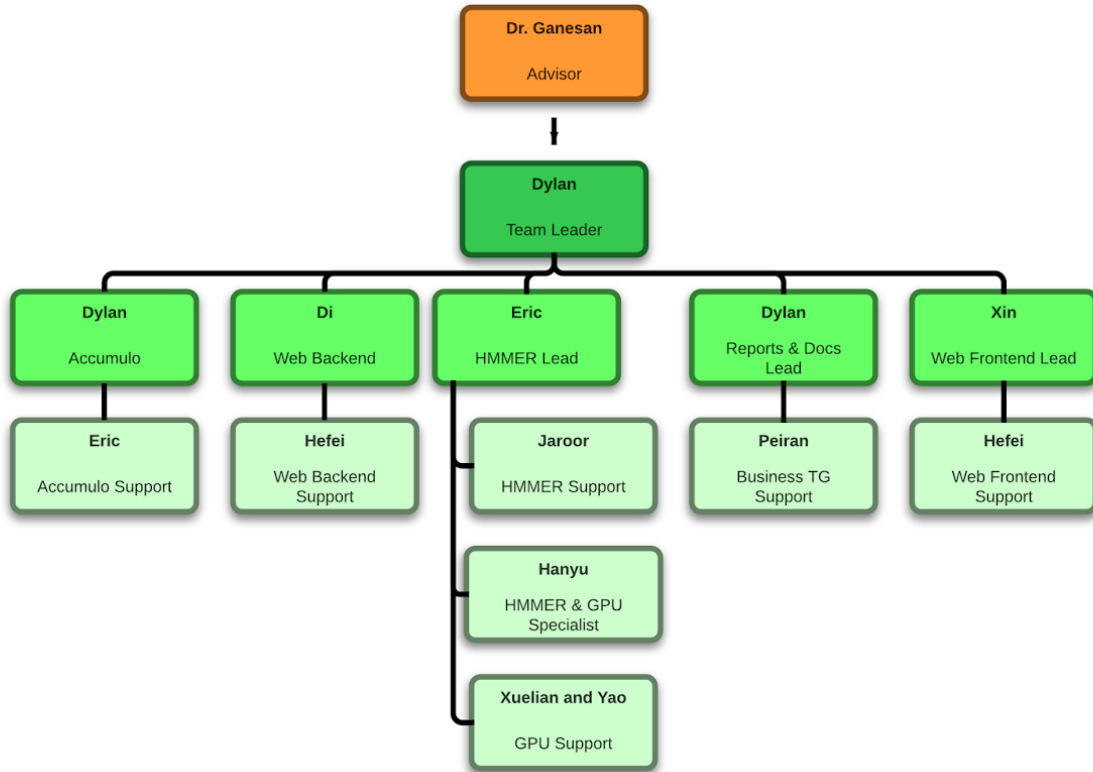
IV.1 Conclusions

IV.2 Recommendations

We recommend any group actively working with Accumulo hire a full-time system administrator. Accumulo runs atop Hadoop and Zookeeper, all of which are nontrivial to set up. The workload compounds when one considers multiple systems networked together. On top of initial setup, the Accumulo stack has many configurable parameters, such as the compaction ratio that trades ingest for query performance. Tuning those is painstaking but necessary for top performance. A system administrator can also handle the mundane tasks of user management, especially if we need to enforce access control / user permissions. As implemented in the project, we configured Accumulo as open, hassle-free and unsecured as possible to speed development.

Appendices

A Team organization chart



D Design Documents: Drawings, Layouts, Analysis reports

Our code is online here: <https://github.com/Stevens-GraphGroup/>

We invite interested readers to view our code on Github rather than kill 50 trees to view it here.

E Team Logistics Systems

We use several services to communicate and store information:

- Github - repository to share code and the website
- Todoist - online task manager for productivity
- Google Drive - file storage and synchronization, collaborative editing
- PuTTY and WinSCP - SSH and SFTP (communicating with remote servers)
- IntelliJ IDEA - IDE with great Java support
- Git - version control system to track, branch and merge changes
- Todoist and FollowUpThen - task managements and reminder systems
- LucidChart - charting tool used to create the Team Organization Chart

F References cited

1 "Sqrrl Enterprise - Linked Data Analysis for Hadoop." *Sqrrl*. N.p., n.d. Web. 30 Nov. 2014. <<http://sqrrl.com/product/sqrrl-enterprise/>>.

- A general data analytics platform, powered by Apache Accumulo. Genetic sequence alignment is possible with Sqrrl Enterprise. HBaaS offers a specialized solution for bioinformatics problems instead. Sqrrl's main customer base uses its cell-level security emphasis for Accumulo.

2 "ArgyleDB Simplifies Real-time Fraud and Security Analytics." *Argyle Data Realtime Analytics for Big Data*. N.p., n.d. Web. 30 Nov. 2014. <<http://www.argyledata.com/product/>>.

- Similar to Sqrrl Enterprise, ArgyleDB is a general data analytics platform, with an emphasis on fraud data.

3 Bilofsky, Howard S, and Burks Christian. "The GenBank® genetic sequence data bank." *Nucleic acids research* 16.5 (1988): 1861-1863.

- Genbank is our primary data source of protein sequence data. We will expand to other sources as time permits.

4 "Basic Local Alignment Search Tool." *BLAST*:. N.p., n.d. Web. 30 Nov. 2014. <<http://blast.st-va.ncbi.nlm.nih.gov/Blast.cgi>>.

- A website that gives researchers the ability to compare amino acid sequences to ones stored in the database.

5 Finn, Robert D, Jody Clements, and Sean R Eddy. "HMMER web server: interactive sequence similarity searching." *Nucleic acids research* (2011): gkr367.

- Introduces the web server delivering HMMER as a service. Uses hidden markov models to align protein sequences. It aims to be more accurate than older scoring methods such as BLAST. We base a great deal of our design on HMMER and use it for performance comparison.

6 "Apache Accumulo." *Apache Accumulo*. N.p., n.d. Web. 30 Nov. 2014. <<https://accumulo.apache.org/>>.

- A high performance database at the center of our design.

7 Bustamam, A.; Burrage, Kevin; Hamilton, N.A., "Fast Parallel Markov Clustering in Bioinformatics Using Massively Parallel Graphics Processing Unit Computing," *Parallel and Distributed Methods in Verification, 2010 Ninth International Workshop on, and High Performance Computational Systems Biology, Second International Workshop on* , vol., no., pp.116,125, Sept. 30 2010-Oct. 1

- As Burrage et al sped up biological computation with GPUs, we aim to do the same on protein sequence computations.

8 Prić, Andreas et al. "BioJava: an open-source framework for bioinformatics in 2012." *Bioinformatics* 28.20 (2012): 2693-2695.

- We use BioJava for FASTA file parsing. BioJava has components for computing sequence alignments, 3D models and much more that we do not use.

9 BioSQL. <http://www.biosql.org/>

- BioSQL is a SQL-based database for storing sequences, features, annotations, etc. We use Accumulo instead.

10 Hung, Che-Lun, and Yaw-Ling Lin. "Implementation of a parallel protein structure alignment service on cloud." *International journal of genomics* 2013 (2013).

- A web service that provides sequence alignments on demand using Hadoop MapReduce in the back end. We expect MapReduce is a better solution for the use case of running alignments against an entire database, as opposed to a selected subset of a database.

11 Kepner, Jeremy et al. "D4M 2.0 schema: A general purpose high performance schema for the Accumulo database." *High Performance Extreme Computing Conference (HPEC), 2013 IEEE* 10 Sep. 2013: 1-6.

- The D4M Schema inspired our database table schema. It is a generic schema template that allows for high flexibility while maintaining decent performance.

12 Michelle Magrane and UniProt Consortium "UniProt Knowledgebase: a hub of integrated protein data" *Oxford Journals*. N.p., n.d. Web. 24 Nov. 2010. <<http://database.oxfordjournals.org/content/2011/bar009.full.pdf+html>>.

- Discusses prowess of UniProt KB as a powerful protein Markov Model database. Also reveals how UniProt KB allows for taxonomic restriction of search and how data is added to the database. UniProt KB is the database we will examine after Genbank.

13 Pruitt, Kim D., Tatiana Tatusova, and Donna R. Maglott. "NCBI Reference Sequences (RefSeq): A Curated Non-redundant Sequence Database of Genomes, Transcripts and Proteins." National Center for Biotechnology Information. U.S. National Library of Medicine, 27 Nov. 2006. Web. 02 Dec. 2014. <<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1716718/>>

- The NCBI RefSeq database is a non-redundant database that we can use as a possible road for future expansion.

14 "BioBench: A benchmark suite of bioinformatics applications." K. Albayraktaroglu, A. Jaleel, X. Wu, M. Franklin, B. Jacob, C.-W. Tseng, and D. Yeung. Proc. 2005 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2005), pp. 2-9. Austin TX, March 2005

- We intend to use similar benchmarking tests as BioBench.

Conference and Journals of Interest for publishing results

- IEEE Transactions on Parallel and Distributed Systems (TPDS). <http://www.computer.org/portal/web/tpds>
- IEEE High Performance and Extreme Computing (HPEC) Conference. <http://www.ieee-hpec.org/>
- Supercomputing Conference (SC). <http://www.supercomp.org/>

G Example Data

gbpri1.fsa_aa -- FASTA file from NCBI GENBANK containing protein sequences

```
>gj|1809232|gb|AAB46355.1| coded for by human cDNA N49626 (NID:g1190792) [Homo sapiens]
MFSSSFLDKKLYVSRGSASTSLPNETLSELETPGKYSLTPPNHWGHPHRYLQHL
>gj|2078460|gb|AAB54048.1| Rod transducin (alpha-1 subunit) [Homo sapiens]
MGAGASAEKHSRELEKKLKEDAEDARTVKLLLLGAGESGKSTIVKQMKIIHQDGYSL EECLEFIIY
GNTLQSI LAIVRAMTTLN IQYGDSARQDDARKLMHMADTIEEGTMPKEMSDIIQRLWKDSGIQACFERAS
EYQLNDSAGYYLSDLERLVTPGYVPT EQDVLRSRVKTTGIIETQFSFKDLNFRMFVGGQRSERKKWIHC
FEGVTCIIFIAALSAYDMVLVEDDEVNRMHESLHFLNSICNHRYFATT SIVLFLNKKDVFF EKIKKAHLS
ICFPDYDGPNTYEDAGNYIKVQFLELNMRRDVKEIYSHMTCATD TQNVKVFVDAVTDIIKENLKDCGLF
>gj|561733|gb|AAB63305.1| MHC class II DRA [Macaca mulatta]
MAESGVPVLGFFIIAVLMSAQESWAIKEEHVIIQAEFYLNPDQSGEFMFDFDGD EIFHVDMAKKETVWRL
EEFGRFASF EAQ GALANIAVDKANLEIMTKRSNNPITNVPPEVT VLTNSPVELGEPNVLICFIDKFSP
VVKVTWLKNGKPVTTGVSETVFLPREDHLFRKFHYLPFLPSTEDIYDCKVEHWCLDAPLLKHWEFDAPSP
LPETTENVV CALGLIVGLVGIIVGT VFIKGVKRSNAAERRGPL
```

globins4.hmm -- example motif Hidden Markov Model file

```

HMMER3/b [3.0 | March 2010]
NAME globins4
LENG 149
ALPH amino
RF no
CS no
MAP yes
DATE Sun Mar 28 09:50:46 2010
NSEQ 4
EFFN 0.964844
CKSUM 2027839109
STATS LOCAL MSV -9.9014 0.70957
STATS LOCAL VITERBI -10.7224 0.70957
STATS LOCAL FORWARD -4.1637 0.70957
HMM A C D E F G H I K L M N P Q R S T V W Y
m->m m->i m->d i->m i->i d->m d->d
COMPO 2.36553 4.52577 2.96709 2.70473 3.20818 3.02239 3.41069 2.90041 2.55332 2.35210 3.67329 3.19812 3.45595 3.16091 3.07934 2.66722 2.85475
2.56965 4.55393 3.62921
2.68640 4.42247 2.77497 2.73145 3.46376 2.40504 3.72516 3.29302 2.67763 2.69377 4.24712 2.90369 2.73719 3.18168 2.89823 2.37879 2.77497 2.98431
4.58499 3.61525
0.57544 1.78073 1.31293 1.75577 0.18968 0.00000 *
1 1.70038 4.17733 3.76164 3.36686 3.72281 3.29583 4.27570 2.40482 3.29230 2.54324 3.63799 3.55099 3.93183 3.61602 3.56580 2.71897 2.84104 1.67328
5.32720 4.10031 9 --
2.68618 4.42225 2.77519 2.73123 3.46354 2.40513 3.72494 3.29354 2.67741 2.69355 4.24690 2.90347 2.73739 3.18146 2.89801 2.37887 2.77519 2.98518
4.58477 3.61503
0.03156 3.86736 4.58970 0.61958 0.77255 0.34406 1.23405
2 2.62748 4.47174 3.31917 2.82619 3.63815 3.49607 2.75382 3.03401 2.75280 2.74783 3.65114 3.24714 2.62341 3.12082 3.11124 2.79244 2.89355 1.88003
5.06315 3.77128 10 --
2.68618 4.42225 2.77519 2.73123 3.46354 2.40513 3.72494 3.29354 2.67741 2.69355 4.24690 2.90347 2.73739 3.18146 2.89801 2.37887 2.77519 2.98518
4.58477 3.61503
0.02321 4.17053 4.89288 0.61958 0.77255 0.48576 0.95510

```

globins4.out -- Example scores for sequences against a motif HMM, followed by alignments

Scores for complete sequences (score includes all domains):

```

--- full sequence --- --- best 1 domain --- #-dom-
E-value score bias E-value score bias exp N Sequence Description
-----
6e-65 222.7 3.2 6.7e-65 222.6 2.2 1.0 1 sp|P02185|MYG_PHYCA Myoglobin OS=Physeter catodon GN=MB PE
3.1e-63 217.2 0.1 3.4e-63 217.0 0.0 1.0 1 sp|P02024|HBB_GORGO Hemoglobin subunit beta OS=Gorilla gor
4.5e-63 216.6 0.0 5e-63 216.5 0.0 1.0 1 sp|P68871|HBB_HUMAN Hemoglobin subunit beta OS=Homo sapien

```

...

Domain annotation for each sequence (and alignments):

```

>> sp|P02185|MYG_PHYCA Myoglobin OS=Physeter catodon GN=MB PE=1 SV=2
# score bias c-Evalue i-Evalue hmmfrom hmm to alifrom ali to envfrom env to acc
-----
1 | 222.6 2.2 1.4e-67 6.7e-65 2 149 .] 2 148 .. 1 148 [. 0.99

```

Alignments for each domain:

```

== domain 1 score: 222.6 bits; conditional E-value: 1.4e-67
globins4 2 vLseaktkvkavWakveadvvesGadiLvrfkstPatqeffFekFkdLstedelkksadvkkHggkvldAlsdalakdekleakld 91
vLse+e++ v++vWakveadv+++G+diL+rifks+P+t+e+F++Fk+L+te+e+k+s+d+kHhg++vI+Al+++I+k +++ea+l+k+L
sp|P02185|MYG_PHYCA 2
VLSEGEWQLVLHVWAKVEADVAGHGQDILRILFKSHPETLEKFDLRFKHLKTEAEMKASEDLKKGVTVLTALGAILKK-KGHHEAELKPL 90
g*****.gg***** PP

globins4 92 selHakkikvdpyfkiIsevlvdvIaIrpkeftadvqaaleKilalvklaskYk 149
+++Ha+k+k+++ky+++se+++vI+r+p++f+ad+q+a++K+l+l++k++a+kYk
sp|P02185|MYG_PHYCA 91 AQSHATKHKIPIKYLEFISEAIHVLHSRHPGDFGADAGQAMNKALELFRKDIAAKYK 148
*****7 PP

```

>> sp|P02024|HBB_GORGO Hemoglobin subunit beta OS=Gorilla gorilla GN=HBB PE=1 SV=2

```

# score bias c-Evalue i-Evalue hmmfrom hmm to alifrom ali to envfrom env to acc
-----
1 | 217.0 0.0 7.4e-66 3.4e-63 1 149 [] 2 147 .] 2 147 .] 0.99

```

Alignments for each domain:

```

== domain 1 score: 217.0 bits; conditional E-value: 7.4e-66
globins4 1 vLseaktkvkavWakveadvvesGadiLvrfkstPatqeffFekFkdLstedelkksadvkkHggkvldAlsdalakdekleakld 90

```


v+L+++ek++v+a+W+kv +v+e+G+++L rl+++P+tq+rFe+F+dLst+d+++++vk+Hgkvl+A+sd+la+ld +l+++++
sp|P02024|HBB_GORGO 2
VHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVYPWTQRRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLN-LKGT
69*****..*****.***** PP

globins4 91 LselHakkikvdpyfklseivdvlaarlpkeftadvqaaleKllaivaklaskYk 149
LselH++kl+vdp++fkl++vlv+vla+++keft++vqaa++K++a va++la+kY+
sp|P02024|HBB_GORGO 89 LSELHCDKLVDPENFKLLGNLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH 147
*****7 PP