DNAvisualization.org: an entirely serverless web tool for DNA sequence visualization

Benjamin D. Lee[*1,2,3], Michael A. Timony[2,4], and Pablo Ruiz[3]

[1]In-Q-Tel Lab41
[2]Harvard Medical School
[3]School of Engineering and Applied Sciences, Harvard University
[4]SBGrid

[*] Correspondence: Benjamin D. Lee <benjamindlee@me.com>

Abstract

Raw DNA sequences contain an immense amount of meaningful biological information. However, these sequences are hard for humans to intuitively interpret. To solve this problem, a number of methods have been proposed to transform DNA sequences into two-dimensional visualizations. DNAvisualization.org implements several of these methods in a cost effective and performant manner via a novel, entirely serverless architecture. By taking advantage of recent developments in serverless parallel computing and selective data retrieval, the website is able to offer users the ability to visualize up to thirty 4.5 Mbp DNA sequences simultaneously using one of five supported methods and to export these visualizations in a variety publication-ready formats.

Introduction

As DNA sequencing technology becomes more commonplace, tools for the analysis of its data are among the most cited papers in science (1). The reason is simple: DNA sequences are, by themselves, almost completely unintelligible to humans. Seeing meaningful patterns in DNA sequences (which are often too large to be shown in their entirety on a screen) is a significant challenge for researchers. One approach to addressing this problem is to convert DNA sequences into two-dimensional visualizations that capture some part of the biological information contained within them. This approach has the benefit of taking advantage of the highly developed human visual system, which is capable of tremendous feats of pattern recognition and memory (2).

A variety of methods have been proposed to convert DNA sequences into two dimensional visualizations (3–12). These methods are highly heterogenous, but,

for the sake of this paper, we will only discuss methods with no degeneracy, *i.e.* methods that produce visualizations which may be unambiguously transformed back into the DNA sequences from which they were generated. All of these methods operate on a single underlying principle: they map each nucleotide in a DNA sequence to one or more points in the Cartesian plane and treat each sequence as a walk between these points.

One effect of mapping each base to at least one point is that the number of points grows linearly with the length of the DNA sequence. This poses a technological challenge, as the ability to sequence DNA has vastly outpaced tools to visualize it. Indeed, there is currently a dearth of DNA visualization tools capable of implementing the variety of methods that have been introduced in the literature (11, 13, 14). Taking inspiration from DNAsonification.org (15), which allows for the auditory inspection of DNA sequences, we propose DNAvisualization.org to fill this gap in the web-based visualization toolset.

Methods and Results

Interface

The user interface for the tool is deliberately simple. A user first selects a visualization method from one of the five currently supported methods (6, 7, 11, 16) and then provides FASTA-formatted sequence data to visualize, either by using the operating system's file-input prompt, dragging-and-dropping files onto the browser window, or pasting the raw data into a text prompt. Upon receipt of sequence data, a loading spinner indicates that the system is processing the data. After the data processing is complete, the loading spinner is replaced with the two-dimensional visualization.

The initial view is such that the entirety of each sequence's visualization is visible: every part of every sequence can be seen. This poses an immediate challenge, as comparing sequences of vastly different lengths will result in the smaller sequence being so small as to be essentially invisible. To solve this problem, the tool allows users to toggle the visibility of sequences by clicking on the corresponding legend entry, which will automatically rescale the visualization's axes to fit the displayed sequences. The legend coloring is dynamic as well. The user may decide to color code the legend either with each sequence (shown in fig. 1a) or each file in its own color (shown in fig. 1b) and toggle between options after the data has been plotted, allowing for both inter- and intra-file comparisons.

To inspect a region of the visualization more closely, a user may click and drag over it to zoom in. When zooming in, a more detailed visualization is shown by asynchronously retrieving data for the region, allowing for base-pair resolution analysis. With a single click, the axis scaling may be reset to the default zoom level.

The title and subtitle of the visualization are dynamically set but may be overridden at any time by the user. If the user wishes, their visualization may be downloaded in one of several formats suitable for publication such as SVG, PDF, JPG, and PNG.

Implementation

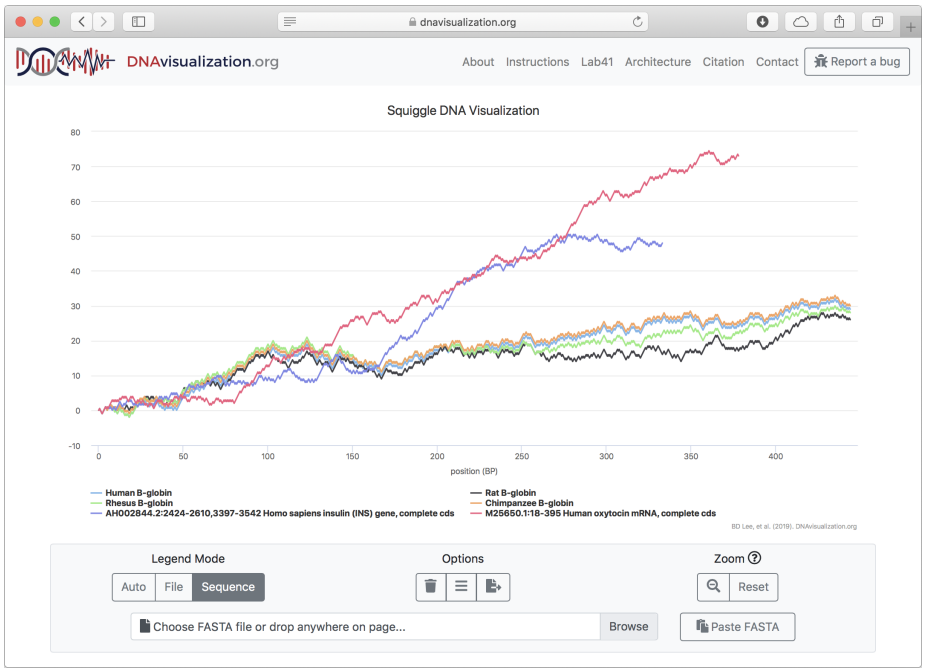
The web tool is built using a novel, entirely serverless architecture, with computing, as well as data storage and selective retrieval, done in a serverless manner. To understand how this system differs from a traditional architecture, consider a traditional approach to building the DNAvisualization.org tool. A server, usually running Linux or Microsoft Windows, is established to handle HTTP requests to the website. This server is either maintained by a university or, increasingly often, a cloud service provider. If there are no requests (as can be expected to be a nontrivial fraction of the time for low-traffic web tools), the server sits idle. When requests are submitted, the server responds to each one. If the server is at capacity, requests may go unanswered or, with additional complexity, more servers may be requested from cloud services provider to meet the greater demand. Data storage is usually provided by a relational database management system (RDBMS), which must also be running on a server.

This paradigm has several disadvantages: disruptions to the server result in disruptions to the website, greater expertise is required for the development and maintenance of the website, the server wastes resources while sitting idle, and the server's computational and storage capacity is directly limited by its hardware.

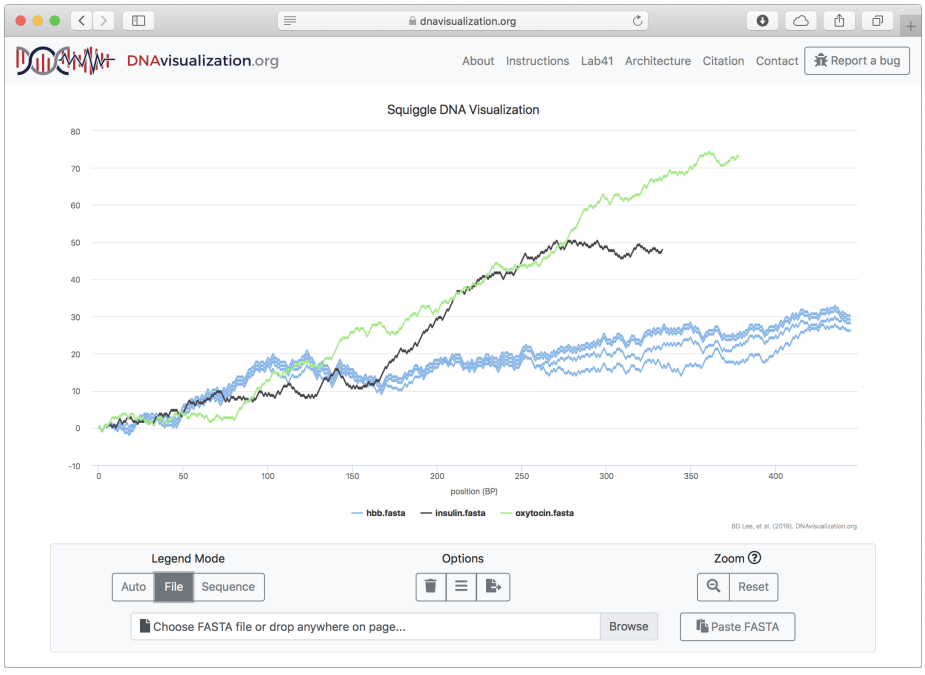
To solve these problems, a new model has been introduced called serverless computing or Function-as-a-Service (FaaS). The basic idea is that a software developer specifies code to be executed (*i.e.* a function) and then invokes it on varying inputs. The cloud service provider is thereby delegated the responsibility for the execution of the code. In this model, the pricing is by function invocation. When not being used, there is no cost to the user. On the other hand, if there are numerous simultaneous function invocations, each invocation is handled separately, in parallel.

By making the serverless function a virtual "server" and invoking the function upon each individual request, one is able to take full advantage of serverless computing. For each request to the website, a virtual server is created for just long enough to respond to the request and then immediately extinguished. This results in the website being able to instantly scale to use precisely the resources needed to meet demand.

DNAvisualization.org is built atop Amazon Web Services (AWS) due to their generous free tier that, at the time of this writing, allows for one-million free function invocations per month using their Lambda serverless compute platform,



(a) Sequence mode



(b) File mode

Figure 1: DNAvisualization.org supports color coding each sequence or file individually.

which is anticipated to easily meet the demand for the site. In the event that the free tier is exceeded, AWS Lambda's pricing is very affordable.

For DNAvisualization.org, we use AWS Lambda to serverlessly transform submitted DNA sequences into their visualizations in parallel, in addition to serving the static assets (*i.e.* HTML, Javascript, and CSS files) to the user. The site uses Python's Flask web framework and has its deployment to AWS Lambda seamlessly automated by the Zappa tool.

It must be noted that using a serverless architecture to host a website is not novel by itself. Rather, the novelty of the architecture lies in its combination of serverless computing for request handling with query-in-place data retrieval on compressed data. As mentioned previously, a normal web architecture would use a server running a RDBMS to handle data storage. In the case of DNA visualization, the database would be used to persist the transformed DNA sequences as x- and y-coordinates that may be queried when zooming in on a region. However, using a database server creates many of the same issues as using a server for web hosting, such as scalability, cost, and parallelism. Instead of using an RDBMS, we used the S3 cloud storage platform combined with the S3 Select query-in-place functionality offered by AWS. In essence, this service allows one to upload a compressed tabular file to S3 and then submit a SQL query to be executed against the tabular data. In this paradigm, pricing is based on the amount and duration of data storage, the amount of scanned during querying, and the amount of data returned by query.

For DNAvisualization.org, each submitted sequence's transformation is stored on AWS S3 in the open-source Apache Parquet tabular data format using Snappy columnar compression. Then, when a user zooms in on a region, a request is sent to AWS Lambda, which submits a SQL query to S3 Select, which in turn scans the file for data in the region. The matching data is then returned to the Lambda function, which downsamples the data if necessary (to prevent wasting users' memory with more data points than can be shown) and returns it to the browser, which then updates the visualization. This process happens entirely in parallel for each sequence that the user has submitted, regardless of how much demand there is on the website, showcasing the usefulness of serverless computing. The S3 buckets (*i.e.* folders) containing the cached DNA sequence transformations are configured such that twenty-four hours after a user has submitted a sequence for visualization, its transformation is automatically deleted, thereby further reducing the cost of the website's operation.

An overview of the architecture is presented in fig. 2.

Discussion

Because DNA sequence transformation is an inherently parallelizable task, the use of serverless computing is a natural fit for this application. However, not all

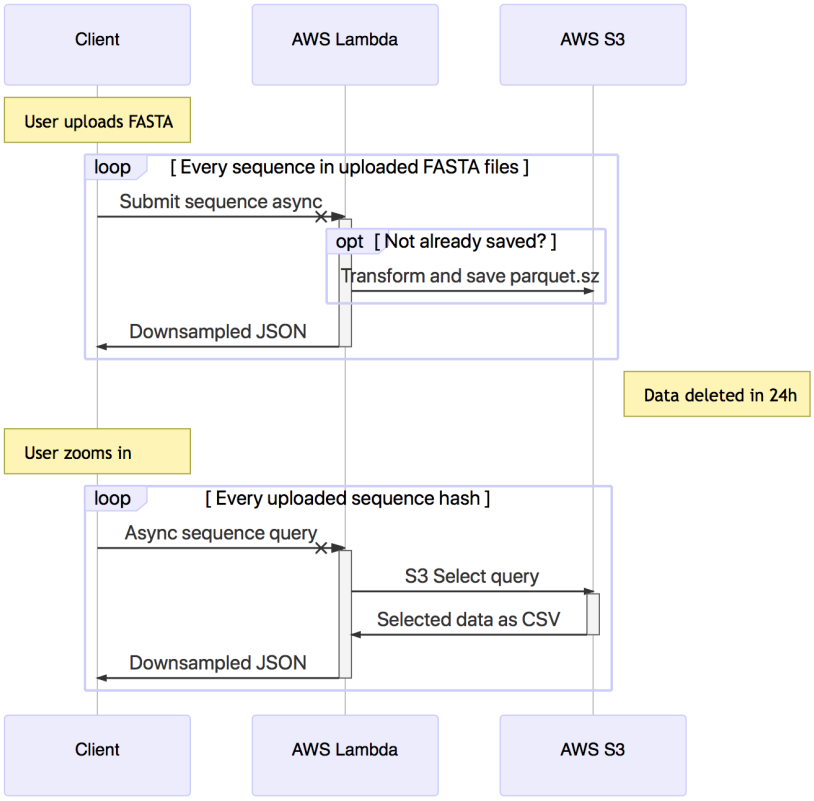


Figure 2: A sequence diagram demonstrating the interactions between the client's browser, AWS Lambda, and AWS S3. There are two sets of interactions: initial sequence transformation and sequence querying. Each of these interactions happens in parallel for each sequence.

web applications for biology are amenable to serverless computing.

The primary limitation of this architecture is necessity for a short duration of computation (currently on the scale of seconds) or, failing that, the ability to parallelize the computation and the data. In addition, memory constraints on the scale of megabytes to several gigabytes must also be respected. Applications which violate these requirements will need significant modifications to this architecture in order to function. As the capabilities of serverless computing increase, the burden of these limitations will decrease. For more information about the limitations of serverless computing, see (17).

These limitations were bypassed by this tool in several ways, which may be of interest to readers attempting to implement similar architectures in the future. When implementing parallelization, we were faced with a choice between higher, file-level parallelization (parsing and transforming each file's sequences in a separate Lambda function invocation) and lower, sequence-level parallelization (parsing the files in the browser and invoking a Lambda function to transform each sequence individually). We initially chose the former but quickly ran into memory issues, even when opting to use the most generous memory allocation available (3,008 MB at the time of writing[1]). To reduce memory demands, we switched to sequence-level parallelism and eliminated as many dependencies as possible.

Currently, the website is limited to visualizing up to thirty sequences of up to 4.5 Mbp each for a grand total of 135 Mbp of sequence data at a time. The total sequence count limitation ensures that our chart renderer can handle rendering all of the points (downsampled to a static 1,000 points per sequence) and the sequence length limitation ensures that the transforming Lambda function's memory is not overwhelmed. In the future, we aim to increase this limit by taking advantage of further optimizations in memory management during transformation and increases in the total amount of memory available to function invocations.

Conclusion

This web tool serves as a demonstration of the applicability of serverless computing to computational molecular biology as well as a useful tool to quickly gain an intuitive visual overview of DNA sequences. While not all applications are amenable to serverless computing, those that are may achieve greater performance with decreased cost and development complexity, a significant advantage over traditional web architectures. By making sequence visualization fast and simple as well as by providing an open-source example of serverless computing and data retrieval, this tool aims to make both of these valuable techniques more widely used within the biological research community.

[1] This total includes all of the function's code as well as the data on which it is invoked.

Data Availability

The website is freely accessible at https://DNAvisualization.org. The software repository is hosted at https://github.com/Benjamin-Lee/DNAvisualization.org.

Funding

This work was supported by the non-profit firm In-Q-Tel, Inc. Funding for open access charge: In-Q-Tel, Inc.

References

1. Wren,J.D. (2016) Bioinformatics programs are 31-fold over-represented among the highest impact scientific papers of the past two decades. *Bioinformatics*, **32**, 2686–2691, DOI: 10.1093/bioinformatics/btw284.

2. Brady,T.F., Konkle,T., Alvarez,G.A. and Oliva,A. (2008) Visual long-term memory has a massive storage capacity for object details. *Proceedings of the National Academy of Sciences*, **105**, 14325–14329, DOI: 10.1073/pnas.0803390105. PMCID: PMC2533687.

3. Randić,M., Vračko,M., Zupan,J. and Novič,M. (2003) Compact 2-D graphical representation of DNA. *Chemical Physics Letters*, **373**, 558–562, DOI: 10.1016/S0009-2614(03)00639-0.

4. Qi,Z.-H., Li,L. and Qi,X.-Q. (2011) Using Huffman coding method to visualize and analyze DNA sequences. *Journal of Computational Chemistry*, **32**, 3233–3240, DOI: 10.1002/jcc.21906.

5. Guo,X. and Nandy,A. (2003) Numerical characterization of DNA sequences in a 2-D graphical representation scheme of low degeneracy. *Chemical Physics Letters*, **369**, 361–366, DOI: 10.1016/S0009-2614(02)02029-8.

6. Yau,S.S.T. (2003) DNA sequence representation without degeneracy. *Nucleic Acids Research*, **31**, 3078–3080, DOI: 10.1093/nar/gkg432.

7. Gates,M.A. (1986) A simple way to look at DNA. *Journal of Theoretical Biology*, **119**, 319–328, DOI: 10.1016/S0022-5193(86)80144-8.

8. Zou,S., Wang,L. and Wang,J. (2014) A 2D graphical representation of the sequences of DNA based on triplets and its application. *EURASIP Journal on Bioinformatics and Systems Biology*, DOI: 10.1186/1687-4153-2014-1. PMCID: PMC3896961.

9. Jeffrey,H.J. (1990) Chaos game representation of gene structure. *Nucleic Acids Research*, **18**, 2163–2170.

10. Peng,C.-K., Buldyrev,S.V., Goldberger,A.L., Havlin,S., Sciortino,F., Simons,M. and Stanley,H.E. (1992) Long-range correlations in nucleotide sequences. *Nature*, **356**, 168–170, DOI: 10.1038/356168a0.

11. Lee,B.D. (2018) Squiggle: A user-friendly two-dimensional DNA sequence visualization tool. *Bioinformatics*, DOI: 10.1093/bioinformatics/bty807.

12. Bari,A.G., Reaz,R., Islam,A.T., Choi,H.-J. and Jeong,B.-S. (2013) Effective Encoding for DNA Sequence Visualization Based on Nucleotide's Ring Structure. *Evolutionary Bioinformatics*, **9**, EBO.S12160, DOI: 10.4137/EBO.S12160. PMCID: PMC3712558.

13. Thomas,J.M., Horspool,D., Brown,G., Tcherepanov,V. and Upton,C. (2007) GraphDNA: A Java program for graphical display of DNA composition analyses. *BMC Bioinformatics*, DOI: 10.1186/1471-2105-8-21. PMCID: PMC1783863.

14. Arakawa,K., Tamaki,S., Kono,N., Kido,N., Ikegami,K., Ogawa,R. and Tomita,M. (2009) Genome Projector: Zoomable genome map with multiple views. *BMC Bioinformatics*, **10**, 31, DOI: 10.1186/1471-2105-10-31. PMCID: PMC2636772.

15. Temple,M.D. (2017) An auditory display tool for DNA sequence analysis. *BMC Bioinformatics*, **18**, DOI: 10.1186/s12859-017-1632-x. PMCID: PMC5404335.

16. Qi,Z. and Qi,X. (2007) Novel 2D graphical representation of DNA sequence based on dual nucleotides. *Chemical Physics Letters*, **440**, 139–144, DOI: 10.1016/j.cplett.2007.03.107.

17. Hellerstein,J.M., Faleiro,J., Gonzalez,J.E., Schleier-Smith,J., Sreekanti,V., Tumanov,A. and Wu,C. (2019) Serverless Computing: One Step Forward, Two Steps Back. In *Conference on Innovative Data Systems Research*. Asilomar, CA.

Figures and Tables Captions

Figure 1. DNAvisualization.org supports color coding each sequence or file individually.

Figure 2. A sequence diagram demonstrating the interactions between the client's browser, AWS Lambda, and AWS S3. There are two sets of interactions: initial sequence transformation and sequence querying. Each of these interactions happens in parallel for each sequence.