Orchestrating Cloud-supported Workspaces for a Computational Biochemistry Course at Large Scale

GIL SPEYER and NEAL WOODBURY, Arizona State University, USA
ARUN NEELICATTU, AARON PETERSON, GREG SCHWIMER, and GEORGE SLESSMAN, CR8DL, Inc., USA

A joint proof-of-concept project between Arizona State University and CR8DL, Inc., deployed a Jupyter-notebook based interface to datacenter resources for a computationally intensive, semester-length biochemistry course project. Facilitated for undergraduate biochemistry students with limited high-performance computing experience, the straightforward interface allowed for large scale computations. As the project progressed, various enhancements were identified and implemented.

Additional Key Words and Phrases: Cloud computing, AlphaFold, Computing education

ACM Reference Format:
Gil Speyer, Neal Woodbury, Arun Neelicattu, Aaron Peterson, Greg Schwimer, and George Slessman. 2023. Orchestrating Cloud-supported Workspaces for a Computational Biochemistry Course at Large Scale . In *Sixth Workshop on Strategies for Enhancing HPC Education and Training (SEHET23) in Practice and Experience in Advanced Research Computing Conference (PEARC '23), July 23–27, 2023, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.XXXX/XXXXXXX.XXXXXXX

1 INTRODUCTION

In early 2022, CR8DL, Inc. launched a new concept in datacenter services, offering easily accessible sandbox services intended to empower large-scale computation. In order to promote this concept with real-world success stories, Arizona State University (ASU) was approached to solicit research or educational computational challenges. A compelling combination of both of these was readily identified: a senior level computational biochemistry course, "Modern Approaches to Biochemical Data Analysis."

The CR8DL resource addressed two critical challenges in the course. First, students would be assigned a semester-long project involving repeated prediction of protein structures from mutated sequences using the Alphafold software [4]. Depending on the length of the sequence, the resulting structure inference could be accelerated employing graphical processing units (GPUs). Second, undergraduate biochemistry students in the course would not have exposure to campus cluster computational resources. To provide services like this for a class would have been expensive for the University in both in terms of time as well as financially (set up, student training, operations staff, time-on-system availability, and so on). Further, the use of a traditional command-line interfaces and use of a job scheduler on University systems pose a steep barrier to entry for a large percentage of the students.

In contrast, CR8DL's services remove these barriers by providing on-demand compute resources that easily fit the purposes of the course. This is provided via a straightforward interface that is easy for students naïve to computer programming to navigate. The Jupyter notebook interface, with its file upload/download capability and persistent pages accessible through a browser, allowed

PEARC '23, July 23–27, 2023, Portland, OR, USA

ACM ISBN 978-1-XXXX-XXXX-X/23/07...$XX.XX
https://doi.org/10.XXXX/XXXXXXX.XXXXXXX

for the students to access these resources with minimal knowledge of the underlying computer system architecture or operating system.

This work emphasizes solely the integrated technologies made available to the students in the course in order to enable an enhanced activity, namely a semester project. No course redesign occurred to accommodate this project. Student information was not used for this study, nor was student feedback solicited.

2 ENABLING HPC TO BIOCHEMISTRY UNDERGRADUATES

For each student, the project entailed starting with a library of one thousand randomly generated sequences of the same size as a target protein domain, AAR10, a plant transcription regulator (Protein Data Base entry 1IRZ) [2]. The goal of the project was to see if they could start from a random library of sequences and create a protein domain with the same alpha carbon backbone structure as the AAR10 domain. They did this via an iterative process. In each cycle they used an algorithm that compared the structure of each generated sequence to the target backbone structure, selected the best structure, mutated it, generated 100 variants of this amino acid sequence, uploaded these to a compute cluster, ran AlphaFold to predict their resulting structure, performed post-processing of these structures involving downloading the results for visual and algorithmic assessment. The cycle was repeated about 10 times. A workflow, showing the iterated steps, including preprocessing and postprocessing steps run on the ASU supercomputer Agave is shown in Figure 1.

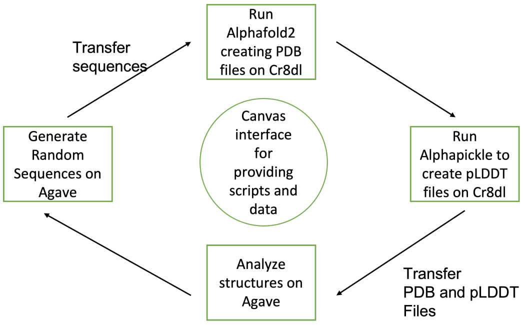

Run Alphafold2 creating PDB files on Cr8dl
Transfer sequences
Canvas interface for providing scripts and data
Generate Random Sequences on Agave
Run Alphapickle to create pLDDT files on Cr8dl
Analyze structures on Agave
Transfer PDB and pLDDT Files

Fig. 1. Workflow diagram, showing the iterated steps.

Throughout this process the student had to make a number of decisions. Not only were they looking for the best fit of the backbone to the target, they were considering the quality of prediction of alphafold as determined by the predicted local distance difference test (pLDDT) score. They also had to decide how many mutations to make each round and then visually decide if the key structural elements were coming into place on the best performers.

By the end of the semester, substantial progress was made by many of the students in using this random mutation followed by selection approach to generating a backbone structure similar to the target. Figure 2 shows the overlap of the target structure (red) and the structure generated after twelve rounds of random mutation followed by selection. The pLDDT was 67.5 (borderline believable) and the root mean squared error (RMSE) between structures was 3.47 based on an angstrom distance scale and focused on the central region ignoring the unstructured portions of the target.

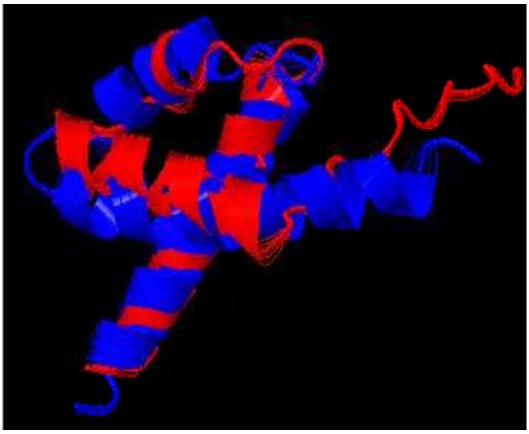


Fig. 2. Overlap between the final alphacarbon backbone structure and the generated structure after twelve rounds of random mutation and selection using Aphafold2 to generate structure data and pLDDT scores. A separate algorithm (in Matlab) was used to generate the RMSE scores. Students used the pLDDT, RMSE and visual inspection of overlapped structures to select which of the 100 mutations in each structure would advance to the next round.

2.1 Compute and Software Resources

The cloud environment provided for this project allowed all students in the class to concurrently share a high-performance compute infrastructure. Resources such as high-capacity storage, CPU, and the latest GPU compute resources were provided. The software components "over the top" of these resources meant that at no time was a user expected to know how to administer or otherwise program on it. The underlying compute infrastructure used for this project consisted of a number of physical "servers" in a cluster configuration. Each user was provided guaranteed, full time access to "slices" of these resources for their own use. Each slice was composed of:

- CPU: 8 cores
- RAM: 32 GB
- GPU: Nvidia A100-SXM4-80G - 1 "MiG" instances, with a "3g.40gb" profile (3 slices 40GB RAM)
- Storage: Unlimited. No user used more that 75GB during the project

These resources were provided on-demand, such that when a user's interaction was completed, they were reclaimed to a pool and made available to other users on the system. At no point was a user denied these resources as there was ample capacity on the entire system for all.

2.2 User access and accounts

Student and instructor access was provided via a web-based interface. Authentication was enabled by way of single sign on (SSO) federation with ASU systems. This allowed users to use their own existing accounts, controlled by ASU's own infrastructure services.

2.3 Input data and executables

The entire AlphaFold database set from the public AlphaFold repository was replicated to a shared storage cluster. These databases were made available to all users via a read-only directory structure represented in their main home directory.

The primary path of job execution was provided as a single Jupyter notebook. Within this notebook, code was placed to automate the job processing. The goal of this code was to abstract much of the complexity from the user, while also leveraging the high-performance compute capabilities of the overall system.

2.4 Workflow

Upon login with a web browser, the users were presented a standard Jupyter Lab interface. Using this interface, users would implement a simple, standardized workflow to launch AlphaFold and calculate the results for further analysis:

(1) Upload fasta files

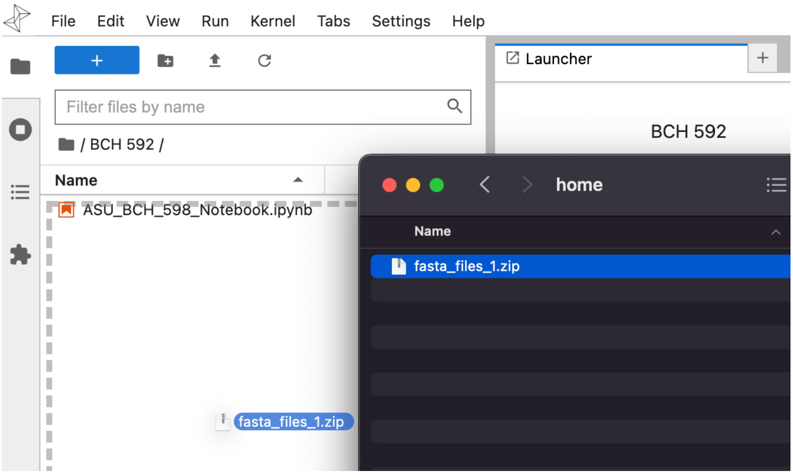


Fig. 3. User uploads their fasta files via a drag and drop interface provided by Jupyter Lab.

(2) Launch the provided Jupyter Notebook and run it
(3) Input the directory of fasta files in the provided notebook interface widget
(4) Select "Run Alphafold"
(5) User collects resulting output for visual analysis.

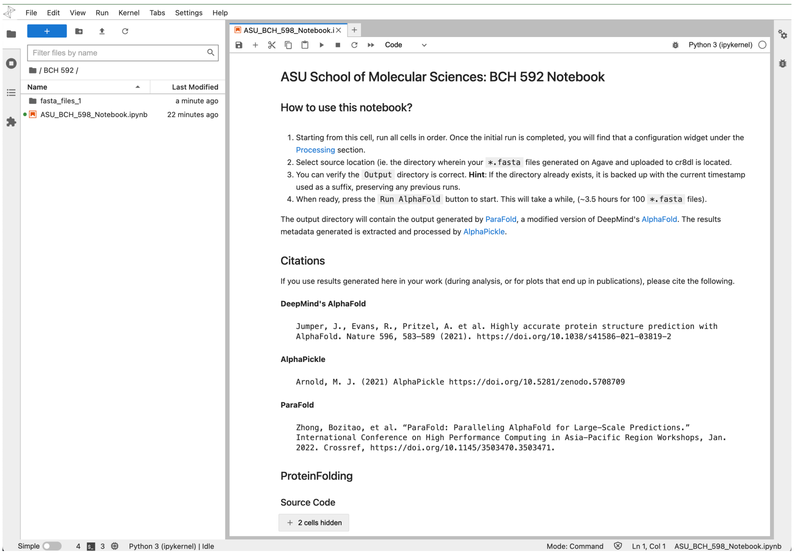
Calculated results were preserved within the user's personal directory for the entire duration of the project, only removed by the user's own actions. The purpose of this was to encourage iterative engagement with the results.

2.5 Output data postprocessing and iteration

Alphafold generates five predicted structures for the input sequence in a pdb format, easily rendered in a viewer. In this way, students could overlay these structures and inspect them visually. Using a Matlab program, RMSE between the atomic positions could be calculated [3]. Finally, the Alphapickle python script provided students with a quantitative and visual means, via pLDDT scores, to assess confidence in output structures [1].

3 REFINEMENTS

Through the semester, the students continued to iterate with the AlphaFold pipeline, uploading new input sets of fasta files representing one hundred new mutations based on the previous runs. As these jobs were run, the CR8DL team tracked performance and investigated opportunities for



ASU School of Molecular Sciences: BCH 592 Notebook

How to use this notebook?

1. Starting from this cell, run all cells in order. Once the initial run is completed, you will find that a configuration widget under the Processing section.
2. Select source location (ie. the directory wherein your `*.fasta` files generated on Agave and uploaded to cr8dl is located.
3. You can verify the `Output` directory is correct. **Hint**: If the directory already exists, it is backed up with the current timestamp used as a suffix, preserving any previous runs.
4. When ready, press the `Run AlphaFold` button to start. This will take a while, (~3.5 hours for 100 `*.fasta` files).

The output directory will contain the output generated by ParaFold, a modified version of DeepMind's AlphaFold. The results metadata generated is extracted and processed by AlphaPickle.

Citations

If you use results generated here in your work (during analysis, or for plots that end up in publications), please cite the following.

DeepMind's AlphaFold

```
Jumper, J., Evans, R., Pritzel, A. et al. Highly accurate protein structure prediction with
AlphaFold. Nature 596, 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2
```

AlphaPickle

```
Arnold, M. J. (2021) AlphaPickle https://doi.org/10.5281/zenodo.5708709
```

ParaFold

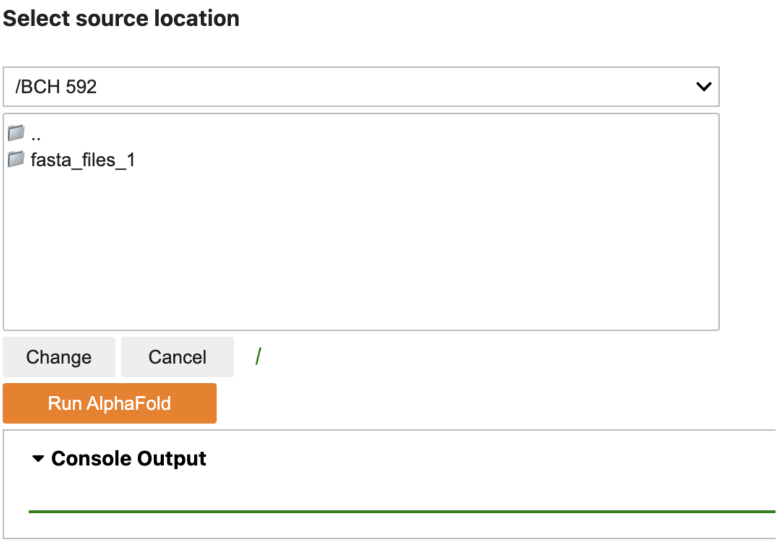
```
Zhong, Bozitao, et al. “ParaFold: Paralleling AlphaFold for Large-Scale Predictions.”
International Conference on High Performance Computing in Asia-Pacific Region Workshops, Jan.
2022. Crossref, https://doi.org/10.1145/3503470.3503471.
```

ProteinFolding

Source Code

+ 2 cells hidden

Fig. 4. Notebook interface.


Select source location

/BCH 592

..
fasta_files_1

Fig. 5. User selects the source fasta file and is then presented with a Run AlphaFold button interface. An output “console” below provides the user with feedback as the job is run.

improvement. Among such refinements were the abstraction of interaction with Python code to a more graphically driven interface as shown in various figures in this document, as well as a restructuring of the way in which the user was expected to interact with the filesystem. This led to a general improvement in workflow, enabling users to be more efficient in their efforts.

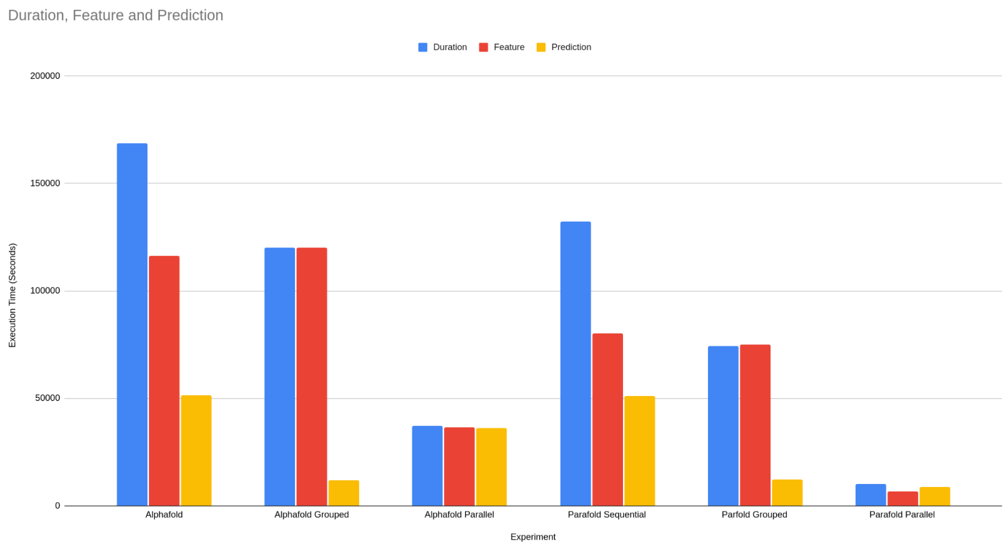


Fig. 6. Alphafold and Parafold job performance comparisons for 100 fasta files across three different processing strategies: "Base", "Grouped", and "Parallel". "Base" is the as-provided, out-of-the-box implementation. The "Grouped" strategy is enhanced by avoiding repeated data loads. "Parallel" execution orchestrates separate, concurrent CPU ("features") and GPU ("prediction") steps. Parafold further exploits maximum parallelism in the feature generation, resulting in improved overall performance. Blue = Overall Duration, Red = Feature Generation Duration, Yellow = Prediction Duration for each job type.

3.1 AlphaFold to ParaFold

Alphafold incurs a time intensive workload, even when presented with a high-performance infrastructure similar to what was provided by the CR8DL team for this project. It was realized early on that by splitting CPU and GPU components of the AlphaFold pipeline, available parallel resources –which are abundant in a datacenter– can be employed to accelerate the workflow. Aptly named, the Parafold utility further parallelizes these computations by exploiting the similarity across the multiple sequences to optimize resources both for the CPU-based feature determination and the GPU-based inference steps [5]. Figure 6 illustrates the enhancements observed for a specific data set, exceeding an order of magnitude acceleration.

4 CONCLUSION AND FUTURE WORK

Continuing efforts are under way to further streamline the user's experience when engaging with complex coursework that leverages high-performance computing resources. User interface improvements, pre-determined visualization outputs, and the ability to run even larger workloads without a need for software development experience are all part of our next iterations. The employment of multiple platforms, despite the vast majority of computation on the CR8DL resource, could be alleviated. Future implementations of the course look to porting tools to remove dependence on the academic Matlab license and co-locating a common folder for project materials for the students.

ACKNOWLEDGEMENTS

The authors acknowledge CR8DL, Inc., and Research Computing at Arizona State University for providing resources that have contributed to the results reported within this paper.

REFERENCES

[1] M. J. Arnold. 2021. *mattarnoldbio/alphapickle: v1.4.1.* https://doi.org/10.5281/zenodo.5752375

[2] Kazuo Hosoda, Etsuko Imamura, Aya abd Katoh, Tomohisa Hatta, Mari Tachiki, Hisami Yamada, Takeshi Mizuno, and Toshimasa Yamazaki. 2002. Molecular Structure of the GARP Family of Plant Myb-Related DNA Binding Motifs of the Arabidopsis Response Regulators. *Plant Cell* 14, 9 (Sept 2002), 2015–2029. https://doi.org/10.1105/tpc.002733

[3] The MathWorks Inc. 2022. *MATLAB version: 9.13.0 (R2022b).* Natick, Massachusetts, United States. https://www.mathworks.com

[4] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature* 596 (2021), 583 – 589.

[5] Bozitao Zhong, Xiaoming Su, Minhua Wen, Sichen Zuo, Liang Hong, and James Lin. 2021. ParaFold: Paralleling AlphaFold for Large-Scale Predictions. arXiv:2111.06340 [q-bio.BM]