An Interdisciplinary Introduction to High Performance Computing for Undergraduate Programs

Cody Stevens
Wake Forest University
stevca9@wfu.edu

Sean M. Anderson
Wake Forest University
anderss@wfu.edu

Adam Carlson
Wake Forest University
carlsoas@wfu.edu

ABSTRACT

The new strategic framework of Wake Forest University seeks to build and strengthen signature areas of excellence in research, scholarship, and creative work that cross academic and institutional boundaries. To support this initiative, the High Performance Computing (HPC) Team has developed an *Introduction to High Performance Computing* undergraduate course that is accessible to students of all levels and of all academic domains. The objective of this course is to build a curriculum that presents HPC as an essential tool for research and scholarship, enables student-faculty collaboration across all disciplines, and promotes student participation in academic research during their undergraduate studies.

KEYWORDS

High Performance Computing, HPC, Cluster Computing, Pedagogy, Education, Interdisciplinary

1 INTRODUCTION

Wake Forest University (WFU) is an R2 liberal arts institution located in Winston-Salem, NC with an undergraduate population of 5,500 and a graduate population of 1,600 across the Reynolda Campus and Business School programs. WFU follows the teacher-scholar model with a student-faculty ratio of 10:1. There is strong support for undergraduate research and experiential learning programs throughout the University. Undergraduate research is so paramount to the University mission, that WFU has a dedicated center, the Undergraduate Research and Creative Activities (URECA) Center, just for this purpose, that provides internal grants to undergraduate students for research fellowships during the summer with a faculty advisor.

Research at the university is supported by the WFU High Performance Computing (HPC) Facility[13]. The facility's main asset, the Distributed Environment for Academic Computing (DEAC) Cluster, provides 4,000 CPU cores, 20 TB of RAM, and 280 TB of storage to researchers across the University. The DEAC Cluster runs on the Red Hat Enterprise Linux (RHEL) 7 operating system. The 280 TB of storage is provided by a NetApp A300 storage array, and is served over NFS to all compute and login nodes. The DEAC Cluster currently supports over 500 faculty, students, and staff from across 15 different academic departments. Training and support for this resource is provided by us (the HPC Team) and this resource is heavily used throughout the *Introduction to High Performance Computing* course.

https://doi.org/10.22369/issn.2153-4136/15/2/3

2 BACKGROUND

Collaboration between us and the Computer Science (CS) Department first began during the Fall semester of 2017. During this time the CS Department was interested in recruiting a team of students to compete in the Student Cluster Competition (SCC) held at the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC). This competition tasks a team of six undergraduate students to compete in a 48-hour competition where each team designs and builds a small HPC cluster. The team must install and run up to five software applications including benchmarks, visualization programs, and a wide range of scientific packages. We attended weekly classes led by a CS faculty member, provided mentorship to students, and presented on HPC topics relevant to the SCC; for instance, on the Slurm scheduler and resource manager, on cluster hardware and networking, and on software compilation. Through this collaboration, four cohorts of students were accepted to compete at the SCC both in-person (2018, 2019) and remotely (2020, 2021).

While this initial class was successful in preparing students to compete in the annual SCC, it did have some shortcomings in both content and student outcomes:

First, this course was offered at the 300-level within the CS curriculum, with a 200-level prerequisite. This prerequisite meant that any student who wished to enroll in the course and participate in the competition would need to have taken at least two courses in computer programming and one course in data structures and algorithms.

Second, as this course was a 300-level elective course, enrollment primarily consisted of students in their Junior or Senior year. While these students did learn many principles of HPC, there was little opportunity for them to use these skills within research programs before graduating.

Third, with the course's primary focus being to train a team to compete in the SCC, there was an implicit enrollment capacity of six students, which is the maximum team size to compete. Assessment for this course was directly tied to performance in the SCC, and therefore there was a disconnect for any student who potentially enrolled in the course and was not on the SCC team roster in both course content and assessment.

These shortcomings drastically limited the broader impact of this initial undergraduate HPC class. We found that the prerequisite courses did not yield the necessary skills to work in an HPC environment, such as familiarity with the Linux command line and

filesystem. Class time was mostly dedicated to learning these fundamental skills, the basics of job scheduling, and exploring computer hardware. This meant that students were not adequately prepared for the SCC as they did not have enough time to sufficiently learn and comprehend more advanced topics like installing and configuring the scheduler, compiling software from source, and familiarizing themselves with the application workflows. It was from this collaboration that the current *Introduction to High Performance Computing* course was developed, and it is designed to address these shortcomings.

3 COURSE DEVELOPMENT

HPC can often be taught from two different pedagogical approaches. The first is on how to *enable* HPC and focuses on the perspective of computer scientists, computer engineers, and cyberinfrastructure professionals who architect and design the hardware, software, and computing environment to support HPC workloads. The second is on how to *use* HPC as a research tool and focuses on the perspective of researchers from many different scientific domains, such as Physics, Biology, Chemistry, Mathematics, and Computer Science, who use HPC to solve complex real-world problems. When developing this course, our first priority was to make it as accessible to as many students as possible across the University. Other institutions that offer HPC courses do so through their Computer Science or Computing Engineering departments[6, 15, 16], and primarily focus on the first approach described above. In order to expand the use of HPC resources across all departments on campus, creating an interdisciplinary course focusing on the second approach was critical to meeting this objective.

3.1 Course Structure

The *Introduction to High Performance Computing* course is currently offered as a Special Topics course within the CS Department at WFU. It satisfies degree requirements for students pursuing a minor or a Bachelor of Arts in Computer Science, but does not satisfy any degree requirements for the Bachelor of Science program. The course meets twice a week for 50 minutes and students earn two credit hours upon completion. The course is unique within the CS Department as it is taught by all three of us (members of the HPC Team). Each instructor has a different background and experience in HPC, and provides their own perspective to the topics covered in the course. The course is currently offered during the Spring semester where it serves as an on-ramp to any student hoping to pursue summer research; this also coincides with two proposal deadlines – for the URECA Center's Summer Research Fellowship and for the SCC in May. We advocate for students to pursue these two opportunities if they are interested in the content of the course, and work directly with students to match them with a faculty research advisor for a URECA fellowship or a hardware vendor for the SCC proposal. The course then provides the necessary training for students to be successful in either endeavor.

Assessment for the course comes from biweekly projects and two larger midterm and final projects. Each project is associated with a curriculum module within the course, and students must complete the tasks by interacting with the DEAC Cluster and answering supplemental questions. We purposefully avoid timed assessments during class, such as quizzes or exams, as tasks that run in an HPC environment are asynchronous, and it can be unreasonable to expect those tasks to finish within a specific period of time. HPC systems have a constant flux of workloads, and there may be periods where job scheduling can take hours or days. Rather than reserving cluster resources for each class and impacting other research workloads, we believe that treating our students just like researchers gives them a more realistic experience of how to use and interact with a shared resource that is used by hundreds of people daily.

3.2 Course Curriculum

The *Introduction to High Performance Computing* course introduces topics using several modules. Each module is presented to the class through a lecture with slides, a hands-on activity performed in class, and a project that students begin during class but have up to a week to complete outside of class. The WFU HPC facility is heavily used throughout the course and students connect and interact with it using Visual Studio Code. We chose this tool because it is platform agnostic and provides students with the same experience no matter what device they bring to the classroom.

We removed all prerequisites for enrollment in order to make the course as accessible as possible to students from all academic domains. Students are not expected to have any prior programming experience before taking the course. We selected Bash and Python as the main applications (and scripting languages) used in the course due to their ubiquity on modern HPC systems and relevancy in data science. We provide all project code for the students, who interact with them on the DEAC Cluster and learn how to monitor their behavior and performance on a real HPC system. The topics covered in the the Spring 2024 semester are listed in the following sections.

3.2.1 Module 1: The Linux Computing Environment. For many students the concept of logging into a remote server is foreign to them. While they can see output of programs and Linux commands through Visual Studio Code, it can be difficult for new users to grasp that these programs are not running on one's personal laptop or workstation. In terms of distributed computing where jobs are submitted through a batch scheduler, such as Slurm, this can add another layer of abstraction that makes the topic more confusing for students, which is why in this module we have students run all exercises on one of the DEAC Cluster's login nodes.

In this module we primarily focus on basic Linux commands such as `cd`, `ls`, and `mkdir`. We also discuss the Linux filesystem and distinguish between absolute and relative paths. We leverage tools such as bashcrawl[10] and the "Password in a Haystack" challenge provided by the Hands on with Frontier GitHub repository[2]. The project for this module requires the students to also utilize helpful Linux commands such as `wc`, `grep`, and `echo`.

Previous iterations of this module had a deeper focus on Linux file permissions, and covered more Linux commands, such as how to determine a server's hostname and IP address, but this overall seemed irrelevant to the audience at this stage of their introduction to Linux, and this content was removed to streamline the course material.

Figure 1: Cody Stevens describes the architecture of a compute node while students disassemble Cisco B200 servers. Photo credit: Sean M. Anderson.

Figure 2: Adam Carlson highlighting the raised flooring and cabling of the WFU Data Center during a tour with students. Photo credit: Sean M. Anderson.

3.2.2 Module 2: Modulefiles and Environment Variables. The WFU HPC Facility heavily relies on environment modulefiles to manage different versions and flavors of scientific software available to researchers, and is common amongst many HPC Centers across the world. This module reinforces paths in Linux from the previous module, and we have students create temporary and persistent enviornment variables that they will use throughout the rest of the course.

The project for this module creates a bespoke environment for each student, and statically compiles several simple C programs that are named after common dictionary words. Students are tasked with identifying the environment module with their username, and analyze the changes to their environment that the module file makes to find a secret hidden program. In this project students utilize module commands such as `show` and `load`, and Linux commands such as `find` and `which` to determine their secret program.

3.2.3 Module 3: Cluster Components and Hardware. This module focuses on the hardware that composes an HPC Cluster, and what resources are available to user jobs. We discuss the components of a cluster compute node, such as CPU Cores and Memory (RAM). Students disassemble decommissioned Cisco B200 blade servers that were previously compute nodes from the DEAC Cluster to physically see how these components are integrated together in a different form factor from their laptop or a desktop workstation.

One of the highlights of the course is in this module where the students take a tour of the WFU Data Center, and see common elements of a data center, such as a raised flooring. The tour covers topics such as power, networking, and cooling within the data center and students get to see the physical hardware that composes the DEAC Cluster. This entire module helps address the layer of abstraction mentioned before in Module 1, and gives the students a physical sense of what resources are available to them when they are about to submit a job.

3.2.4 Module 4: The Slurm Resource Manager and Scheduler. At this point in the course, students should be familiar with the Linux environment, and should understand the basics of how modulefiles work and the resources available to them on the DEAC Cluster. This module ties all of these components together and covers the basics of submitting a batch job through Slurm using a batch script, and how to monitor a job and the state of a Slurm cluster using the commands `squeue` and `sinfo`.

For this project we utilize the software application Blender to render one of the sample files that we have tweaked slightly to provide some WFU flair. While this application may not be common on a given scientific computing cluster, overall we think this is a great application for the students because the rendering process for a single scene can take up to 16 minutes with a single core, and consumes roughly 8Gb of memory or RAM and produces a nice picture and visual feedback for the students. We utilize these resource requirements to show the students common errors that Slurm will provide when too little time or memory is requested. Students also see a significant speedup when requesting additional CPU cores, and students can see the benefit of allocating more CPU cores to a given Slurm job.

We have also used Blender in class to demonstrate the ability of Slurm Job Arrays by rendering one of the sample movies provided by Blender, where each job with the job array renders a single scene before they are all stitched together to create the final movie. We did not use this example in the most recent iteration of the course due to time constraints, but feel it is worth mentioning as we plan to use it again in future offerings of the course.

We believe that the conclusion of this module marks the completion of what a typical HPC training course for a new user would cover. At this point in the course a student should be able to:

- Login to a login node, and interact with the login node using basic Linux commands to organize and store their data.
- Interact with environment modulefiles to see which software applications are available and load appropriate modulefiles for their research.

- Understand the basic resources a cluster provides for their job, and how to create a Slurm batch script to submit jobs to the cluster.

The largest hurdle we have observed for new users to the DEAC Cluster is learning the basic Linux environment. While this content could be covered in a one to two day workshop, we believe that the weekly projects that all require Linux commands to be executed on the login nodes gives the students more structured experience that they would not otherwise get from a one-on-one training session, or a small workshop. We believe that content up to this point in the course could be offered as a half semester class for any prospective student interested in research, and at this point in the course a student should be able to adequately pursue independent research with a faculty advisor.

It is at this time that we also introduce students to a general HPC workflow using Python and the Palmer Penguins dataset[4]. From this moment on in the course Python will be used more heavily, and this interactive activity gives a good example for how data can be preprocessed before analysis, and provides some context for what future Python programs used in the course may be doing.

3.2.5 Midterm: Asymptotic Complexity. As previously mentioned, the first half of the course could be described as a general training for a prospective researcher. The second half of the course serves as an introduction to many concepts we see in HPC, but may not be relevant to every user on a given research cluster. From this point on all work is submitted through batch submission using Slurm.

In this module we discuss asymptotic complexity using real world examples to describe linear, sub-linear, polynomial, and exponential scaling for computational complexity. We also discuss Amdahl's Law and discuss the consequences of requesting too many resources for a given problem. For the midterm we use a modified version of the color quantization using K-means clustering example with scikit-learn[8, 11]. We provide the students with five images of the same subject with varying pixel dimensions to show the speedup as the size of the image increases. Like the previous application, Blender, utilizing images provides a less abstract approach to the topic and with this example program we observe that the sampling of the image performs at the same or worse speed using multiple processors than the assignment of colors using k-means or random clustering, which decreases in computation time as more processors are added to calculations. What the students ultimately observe is that there are distinct parts of the program, and they each do not scale proportionally to one another as more resources are given to the problem.

Figure 3: 3D Render of the WFU-customized Cozy Bakery[9] scene using Blender.

3.2.6 Module 5: Networking and Storage. Networking and storage are essential components to any HPC environment. We instruct the students that on the DEAC Cluster, there exist three directories the students can actively write and save data to. There is their home directory located under `/home` which serves as their starting point on the cluster, but that it is best practice to store input and write output of their research endeavors to their `RESEARCHPATH` which is a directory that exists on a larger pool of storage that is managed by a quota. On the DEAC Cluster, both of these paths exist on a shared filesystem and are shared via NFS to the login and compute nodes.

In this module we also discuss the concept of latency for data transfers between a shared storage array and the compute and login nodes. We discuss local storage, which is commonly referred to as scratch storage, that is temporary in nature and that in theory provides better performance over reading and writing data over an NFS filesystem. In practice, we see a minimal performance gain using the SSD local scratch directories on the DEAC Cluster versus the NFS filesystem provided by the NetApp A300 storage array, but we still believe this module is important as latency is something that will always persist in the field of HPC.

3.2.7 Module 6: Scientific Software. Compiling scientific software is a critical component of enabling HPC workloads, and in this module students learn about how software is installed and managed in an HPC environment. A simple example of compiling Nyancat CLI[7] using GNU make is done in class, and we discuss different versions of software as well as the performance of software when compiled using different compilers, such as GNU and Intel. We discuss what files are created when software is compiled and installed, and environment variables, such as `PATH` and `LD_LIBRARY_PATH`, that are relevant to using software that is compiled from source.

This module directly ties back to Module 2, and gives a more practical example of why we utilize modulefiles in an HPC environment. For this project the students are tasked with installing Python 3.9 into a scratch directory. Due to the DEAC Cluster having a base compiler of GCC 4.8.5, the students would encounter an error if they did not use a more recent compiler, such as GCC 10.2.0, which they are instructed to use. Students are tasked with installing their Python software into a scratch directory which directly references the content covered in Module 5, and students do this both interactively on a login node as well as on a compute node with a job submitted through Slurm.

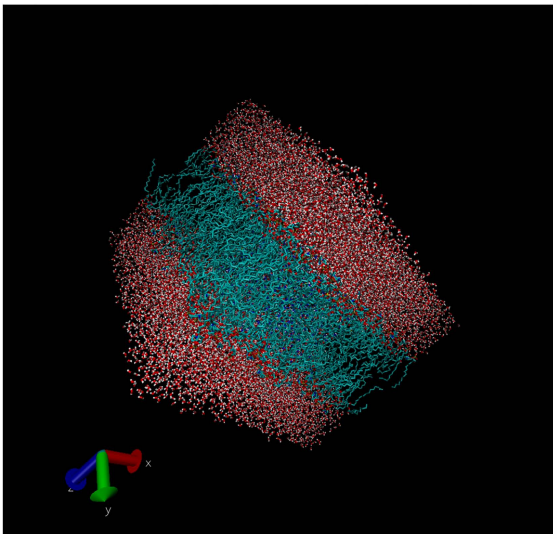

Figure 4: Visualization using VMD of the protein structure output of GROMACS by Cassandra Hung, Spring 2024.

3.2.8 Module 7: GPU Computing. With the rise of Generative Artificial Intelligence (AI) and the prevalence of NVIDIA Data Center GPUs in HPC facilities today, this is a module that is relevant to the current landscape of HPC. This module is split over two different lectures. The first lecture discusses GPU hardware, and the history of the GPU and how it has advanced today up to NVIDIA's latest announcement of the Grace CPU and Blackwell GPU architectures and the GB200. This lecture explains the historical nature of GPU hardware, and how it differs from the CPU and CPU memory. The second lecture focuses on AI, and the software that enables it, such as TensorFlow and PyTorch. We discuss the rise of AI since the early 2010's up to the Generative AI and Large Language Models we see today.

We use this module to introduce the concept of interactive Slurm jobs, as GPU compute nodes have hardware and software that is not available on the DEAC Cluster login nodes, such as the NVIDIA driver. The students for this project deploy Meta's Llama2 13b-chat model[14] using llama.cpp[3]. When compiled this software produces a chat bot the students can interact with and compare the performance to other Generative AI tools, such as ChatGPT. This code is executed on CPU compute nodes with varying numbers of CPU cores, and on the GPU nodes using NVIDIA V100 GPUs.

3.2.9 Module 8: Parallel Frameworks. The last module covered within the class is on parallel frameworks such as message passing and threading. We primarily focus on MPI and OpenMP for each of these frameworks, and we discuss the memory management and use cases for each. Students complete a hands-on activity in teams that tasks them with sorting cards given different requirements. Students are responsible for assigning themselves to different roles in the sorting process as they work together, just as MPI processes and OpenMP threads would interact within the same context, and we observe the real time it takes us as humans to sort the cards using this division of labor.

Table 1: Student enrollment per semester.

Semester	Total Enrollment	Male	Female
Fall 2022	7	6	1
Spring 2022	12	10	2
Spring 2023	16	8	8
Spring 2024	18	9	9

The final project brings both of these concepts together, and tasks the users with installing the molecular dynamics software GROMACS[1] from source and comparing the performance via throughput of a simulation using different combinations of MPI processes and OpenMP threads. Students compare over 50 different combinations using up to 48 CPU cores for a single job, and then produce a visualization of the resulting protein structure using the VMD software package[5].

4 OUTCOMES

Total enrollment in the *Introduction to High Performance Computing* course has grown each semester, with our most recent semester having a total of 18 students. There has been an equal number of men and women enrolled in the course for the previous two semesters, but more than half of all students have been Computer Science majors. This is to be expected, as the course if offered through the CS Department, but this bias needs to be reduced in order to foster more interdisciplinary collaboration. Removing prerequisite classes has made the course more accessible, but there are still many opportunities for more inter-departmental collaborations.

From the Spring 2023 cohort, we had 4 students participate in the Student Cluster Competition (SCC) through the virtual IndySCC competition. Our WFU team, the Daemon Deacons, finished 4th overall in the competition and 1st amongst US teams. One of the software applications for the IndySCC was GROMACS which students already had prior experience with from our course. The final result corroborated that this team was better prepared for the SCC challenges when compared to teams from previous years.

This past semester we saw former students using the WFU HPC Facility for courses in finance, natural language processing, and science-guided machine learning. The WFU HPC Facility supported entire classes on the latter two topics. One student from the finance course had taken our class in the Spring 2023 cohort, and decided

Table 2: Student enrollment by academic major.

Major	Students
Computer Science	27
Mathematics & Statistics	11
Finance	4
Engineering	3
Economics	3
Biology	2
Undeclared	2
Political Science	1

to use the DEAC Cluster to perform a look-back analysis to build the best stock portfolio for her final project. This work was done independently of any other student or faculty involvement and was thanks to their experience in the *Introduction to High Performance Computing* course. From our most recent Spring 2024 cohort, two students have received URECA Research Fellowships for Summer 2024 and will be pursuing research with faculty in the CS Department; one student will be pursuing research opportunities in the Economics Department later this year. We will be working with these students to sponsor travel and accommodations to attend the SC24 conference later this year.

It is this engagement that we hope to foster through our course. By exposing these topics to a diverse student body we can ensure that the HPC resources are well-utilized and continue to contribute to student success throughout their time at WFU.

5 CONCLUSIONS

There is an implicit bias when offering HPC topics from only one academic department – this contradicts the interdisciplinary nature of HPC work. The demand for these resources in higher education will continue to expand as more and more academic disciplines attempt to educate students on the most relevant problems in today's technological and data-driven world. Advances in artificial intelligence and machine learning will necessitate powerful compute resources that can only be provided by HPC facilities. These facilities can also provide students with a unified computing environment and are capable of handling the additional computational requirements of courses that adequately prepare students for employment and post-graduate opportunities. By developing curricula that are accessible to students across all academic disciplines and different skill levels, the concepts of HPC can be introduced early in the undergraduate career path, and can have greater success later as HPC becomes more integrated in other academic programs. This provides greater utilization of HPC facilities, and eases the burden of individual training on faculty and other cyberinfrastructure professionals.

6 FUTURE WORK

The *Introduction to High Performance Computing* course in its current iteration has been offered for four semesters as a Special Topics course within the Department of Computer Science. Our next step is to establish a permanent course reference number within the CS curriculum and expand the number of academic credit hours to 3. With our permanent course listing and our refined curriculum, we intend to publish our projects on GitHub[12] to allow anyone to download each module and project, with bootstrap scripts for popular Linux flavors. Lastly, to break away from any departmental bias and to cater to the true interdisciplinary nature of HPC, we will be working towards creating our own academic program and will collaborate with departments across campus to create accessible projects and coursework for faculty to incorporate into their classes and majors.

ACKNOWLEDGMENTS

Computations were performed using the Wake Forest University (WFU) High Performance Computing Facility, a centrally managed computational resource available to WFU researchers including faculty, staff, students, and collaborators.

The authors would like to thank Dr. William Turkett , Chair of the Department of Computer Science, for fostering this ongoing collaboration, and Dr. Sam Cho for developing the initial High Performance Computing course which laid the foundation for the *Introduction to High Performance Computing* course as it is today.

Special thanks to the Information Systems Leadership Team for supporting the authors in this initiative to bolster the strategic framework of the University and foster a community of learning.

REFERENCES

[1] Mark Abraham, Andrey Alekseenko, Cathrine Bergh, Christian Blau, Eliane Briand, Mahesh Doijade, Stefan Fleischmann, Vytautas Gapsys, Gaurav Garg, Sergey Gorelov, Gilles Gouaillardet, Alan Gray, M Eric Irrgang, Farzaneh Jalalypour, Joe Jordan, Christoph Junghans, Prashanth Kanduri, Sebastian Keller, Carsten Kutzner, Justin A Lemkul, Magnus Lundborg, Pascal Merz, Vedran Miletić, Dmitry Morozov, Szilárd Páll, Roland Schulz, Michael Shirts, Alexey Shvetsov, Bálint Soproni, David van der Spoel, Philip Turner, Carsten Uphoff, Alessandra Villa, Sebastian Wingbermühle, Artem Zhmurov, Paul Bauer, Berk Hess, and Erik Lindahl. 2024. GROMACS 2023.4 Source code.

[2] Oak Ridge Leadership Computing Facility. 2021. Hands on with Frontier. Retrieved February 2024 from https://github.com/olcf/hands-on-with-frontier

[3] Radoslav Gerganov. 2023. LLaMA c++. Retrieved April 2024 from https://github.com/ggerganov/llama.cpp

[4] Kristen B Gorman, Tony D Williams, and William R Fraser. 2014. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus Pygoscelis). *PLoS One* 9, 3 (March 2014), e90081.

[5] William Humphrey, Andrew Dalke, and Klaus Schulten. 1996. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics* 14 (1996), 33–38.

[6] Steve Jones. [n. d.]. ME 344: Introduction to High Performance Computing. https://bulletin.stanford.edu/courses/2191441

[7] K Lange. 2013. Nyancat CLI. Retrieved April 2024 from https://github.com/klange/nyancat

[8] Robert Layton, Olivier Grisel, and Mathieu Blondel. [n. d.]. Color Quantization using K-Means. Retrieved March 2024 from https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html

[9] Nicole Morena. 2023. Cozy Bakery - Blender 3.5 Splash Screen. https://www.artstation.com/nickyblender

[10] notklaatu. 2019. bashcrawl. Retrieved February 2024 from https://gitlab.com/slackermedia/bashcrawl

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[12] Cody Stevens, Sean M. Anderson, and Adam Carlson. 2024. Introduction to High Performance Computing. https://github.com/WFU-HPC/Introduction-to-High-Performance-Computing

[13] Information Systems and Wake Forest University. 2021. WFU High Performance Computing Facility. https://doi.org/10.57682/G13Z-2362

[14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:cs.CL/2307.09288

[15] Iowa State University. [n. d.]. COMS 424: Introduction to High Performance Computing. https://catalog.iastate.edu/azcourses/com_s/

[16] Rich Vuduc, Andrew Becker, Catherine Gamboa, and Amanda Deisler. 2023. CSE 6220: Intro to High-Performance Computing. https://omscs.gatech.edu/cse-6220-intro-high-performance-computing