Visualizing Performance Characteristics of Computations

Basic Information

- **Name**: Freyam Mehta
- **Major**: Computer Science And Engineering
- **University**: International Institute of Information Technology Hyderabad, India
- **Email**: freyam.mehta@students.iiit.ac.in
- **Github and Gitter**: @freyam
- **Slack**: Freyam Mehta
- **Timezone**: Gulf Standard Time (UTC +4:00)

Project Description

Dask already has the `.visualize` method and the `dask.visualize` function, which uses `graphviz` to render a task graph. It is simple and effective. It easily tracks the parallelism between function calls and helps optimize the code further by decorating the function to operate lazily (Delayed object). It becomes effortless for the user to find opportunities to convert a sequential code into a code that can be optimized by finding parallelism between different tasks. It also lays out the function names and the flowchart of the entire computation.

There is a scope of improvement here. The task graph can illustrate more relevant details about the performance of the computation, how each block is contributing to the overall complexity of the program and how some blocks can be easily optimized by simplifying computation and improving parallelism.

Motivation

Rightly pointed out by @jrbourbeau over at https://github.com/dask/dask/issues/7141, the graph doesn't display much information about the **performance characteristics of a task** itself. What it excels at is showing a task graph of computation. It only excels at finding the

layers of the code flow and offers a precise chronology of events.

Moreover, the output of `.visualize` is often overwhelming, especially for data with many nodes. (Surprisingly, even without large numbers of nodes, one can see the low-level graph output from `map_overlap` on an array with two chunks https://github.com/dask/dask/discussions/7404#discussioncomment-511750).

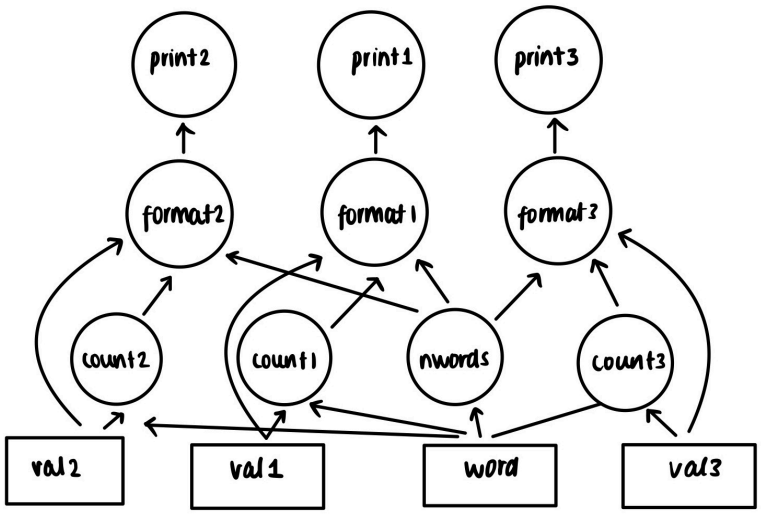
The users need to find areas of possible inefficiency in their complex computations easily, and there's something I can do about this.

Objective

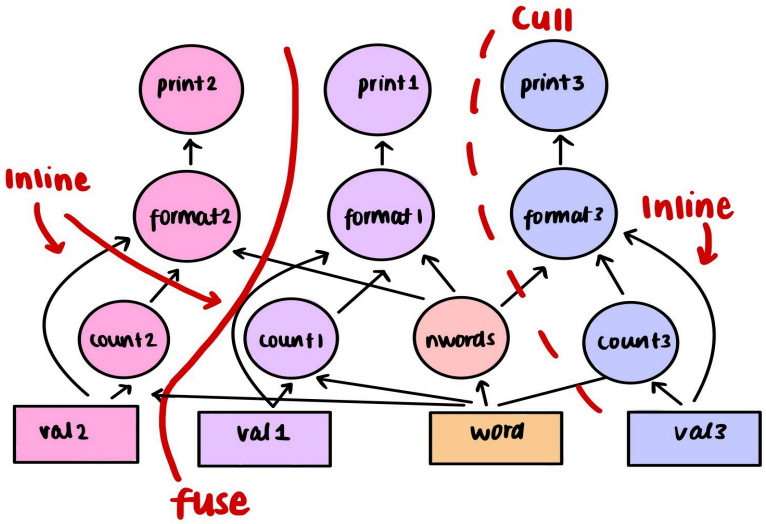
Currently, `.visualize` renders a static black-and-white image showing the blocks and their connections. But we can do so much more with some added flavor!

Feature 1 (Colored Graphs)

To help the user make sense of the performance and the scope of optimization, I will **highlight** the **unnecessary tasks** (`cull`) using dark colors or reducing its transparency while rendering. The same can be done with the **redundant constants** (`inline`). **Linear tasks** flow can be labeled/colored similarly, so it becomes easy to spot room for merging these chains (`fuse`). I will also show a side-by-side comparison of the pre-optimization and the post-optimization graphs and highlight the changes. Also, I will color the **I/O tasks differently from the computational tasks** by keeping a list of some commonly used task names (such as `read_csv`). This allows the user to learn more about the program's performance characteristics.



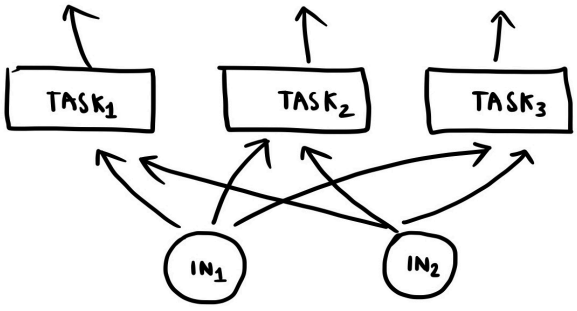
CURRENT VISUALISATION



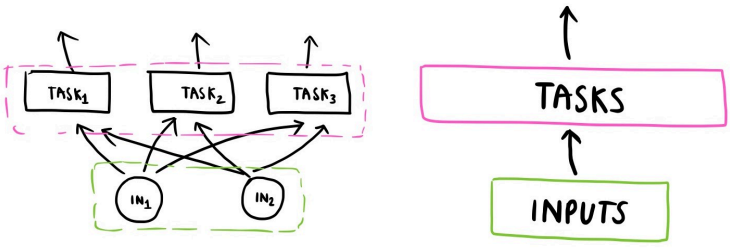
IMPROVED VISUALISATION

Feature 2 (Collapsible Blocks)

To make visualization of large computations easy and visually appealing, I will introduce **collapsible blocks** that **group giant blocks into smaller blocks**, thus **simplifying the rendered graph** and also makes it easier to work with crucial data containing **a large number of nodes**. Currently, graphviz struggles with large computations. Simplification of the circuit on top of the pre-existing `dask.optimization` would make it more feasible (and quicker) to render graphs for large computations. (A foreign library can be considered for this).



CURRENT VISUALISATION



IMPROVED VISUALISATION

Feature 3 (Labelled Metadata)

To get more meaning from the task graph itself, I will **highlight more metadata** (opens up doors for interactivity). I will label **"type of the graph"**, **"the input/output characteristics"**, and **"the number of tasks"** at **block-level** and **layer-level**.

Rightly pointed out by several developers over at https://github.com/dask/dask/issues/7301, the integration of foreign libraries/extensions like `cytoscape`, `d3-graphviz`, and `ipyelk` can help us here. They bring a whole new level of visualization to our `.visualize` by adding new interactive elements that let the user play around with their task graph and provide a better and more robust analysis of the computation.

I will choose one library/extension based on the following factors:

- The **feasibility of implementing the mentioned features** (keeping in mind the extent of stretch goals).
- The **performance chart of the library at a high number of nodes** (using a benchmark script).
- The **community support and the feasibility of the integration of the library**

with Dask.

Deliverables

- A **Jupyter MIME-Extension** or an `ipywidget` Integration, which adds a new flavor of visualization (depending upon the chosen implementation method).
- **High-quality code** with **detailed documentation** (`Sphinx`), **thorough testing**, and **code styling** (`black`).
- A **detailed and well-documented tutorial of all visualization features** will be developed during GSoC to provide developers with reference.
- **Weekly updates regarding the development of the project to the mentors**. This will help bolster my accountability and transparency.

Stretch Goals

Once I am done with the implementation of the mentioned features, I would like to work on the following features to enhance the overall visualization process. I haven't researched much on this, but **I will also keep the following features in mind while choosing a suitable library**.

- [1] **An interactive graph** that lets the user move around task blocks and re-render the graph without changing the source code. The user would be able to alter the program's flow and see how it changes the overall complexity of the computation. Interactivity makes it straightforward for the user to make sense of the computation graph at a higher level and a lower level at the same time.
- [2] **A standalone server** on which the new tool would function. I will first integrate the new features into the Jupyter environment and then work on the Bokeh Server implementation. This would allow me to broaden the usage of the newly created tool. Alternatively, I could somehow devise a new type of format for the graph, which can be manipulated outside of the Jupyter environment.

Project Breakdown

Three well-defined goals that determine project status which will help in the evaluation are as follows:

First Evaluations (July 12 - 16, 2021)

- Implementation of graph coloring using a suitable JS library.
- A blog on the implementation of the first feature.

Final Evaluations (August 16 - 23, 2021)

- Implementation of collapsible blocks and adding more metadata on the graph using a suitable JS library.
- A blog on the implementation of the second and third feature.
- Final notebook demonstration.

Project Timeline

Below is my **timeline** for the completion of the project. This is meant as more of **an outline for what order I will complete the project, rather than a raw week-by-week breakdown** (as in, some weeks may blend into each other). A more detailed schedule (after gaining an in-depth understanding of technologies used) will be planned during the pre-GSoC and community bonding period and shared with the mentor.

Regardless, I will have **at least one deliverable at the end of each week** to be evaluated.

I do not have any other commitments during the summer. I am lucky that the GSoC timeline works perfectly with my course schedule as well; hence, I will not be busy with coursework during the GSoC period, and **I can sincerely commit to the timeline I have illustrated below.**
If something comes up, I will inform my mentor a week before and limit my unavailability to no more than two days. I am willing to adjust and re-plan the timings as per mentor/admin availability and any unforeseen requirements in the future.

Each week, time will be divided (according to workload) amongst **planning, learning, coding, documenting, and testing features**. Except for the developer's guide, all documentation will go **parallelly with the development**. This will help keep a profound grasp of the code implementation and working, minimize bugs in later stages. Weekends will be primarily dedicated to **testing, bug fixing, and blog writing**. Fortnightly blogs will be maintained at

https://medium.com/@freyam and will include highlights of the development process, the methods used to solve issues encountered, and my overall experience. The final blogpost would be up on https://blog.dask.org/.

- **Weekdays** (Monday to Friday): **5 - 6 hours** (**+ 2 hours if needed**)
- **Weekends** (Saturday and Sunday): **4 - 5 hours** (**+ 3 hours if needed**)
- **Cumulative** (Monday to Sunday) **33 - 40 hours** (**+ 16 hours if needed**)

I will be active on **Slack/Email/Gitter** between **7:00 AM (3:00 AM UTC) and 11:55 PM (7:55 PM UTC)** and be available for communication at all times decided by my mentor and discuss new ideas and methods throughout the project.

Pre GSoC (till 16th May)

- Make more contributions by fixing issues over the Dask ecosystem and adding features to bolster my understanding of the codebase and the scope of the `visualize()` method.
- Get more familiar with the operational workflow of Dask Task Graphs by reading up on https://docs.dask.org/en/latest/graphviz.html.

Community Bonding (17th May to 7th June)

- The main focus in this period will be to **frame a roadmap for the project** with the mentor's guidance (along with improving bonding, which is what the period is for).
- **I will plan the implementation of features and discuss them with the rest of the Dask community beforehand to minimize bugs and improve structure.**
- I will use this period to study the already present ideas on improving `visualize()` and to figure a concrete plan to integrate the new features.
- **I will frame the integration of various parts of the project into one another. I will prepare a source code directory layout to ensure a smooth flow in the later development stages.**

Week 1 (7th June to 13th June)

- Assess all proposed JS frameworks for suitability and conduct experiments with various technologies being used to gain a deeper understanding of the same.
- **Deliverable: Comparison table of the frameworks & top choice**

recommendations for further assessment from mentors.

Week 2 (14th June to 20th June)

- Assess 2-3 top choice JS frameworks.
- **Deliverable: A short demonstration for each, showing how we might pass a dask `graphviz` task graph and display it. Benchmark performance for graphs with large numbers of nodes. Make recommendations for our top choice framework.**

Week 3 (21st June to 27th June)

- Discuss/rank new feature ideas and make a detailed plan for the next six weeks.
- Begin work on Feature 1.
- **Deliverable: A branch with multiple commits showing development progress on Feature 1.**

Week 4 (28th June to 4th July)

- Continue work on Feature 1.
- **Deliverable: A PR for Feature 1 ready for review.**

Week 5 (5th July to 12th July)

- Make changes to Feature 1 based on the feedback.
- **Deliverable: Completed the implementation of Feature 1.**

Week 6 (13th July to 18th July)

- Begin work on Feature 2.
- **Deliverable: A branch with multiple commits showing development progress on Feature 2**

Week 7 (19th July to 25th July)

- Continue work on Feature 2 (priority).
- Begin work on Feature 3.

- **Deliverable: A PR for Feature 2 ready for review.**
- **Deliverable: A branch with multiple commits showing development progress on Feature 3**

Week 8 (26th July to 1st August)

- Make changes to Feature 2 based on the feedback.
- Continue work on Feature 3.
- **Deliverable: Completed the implementation of Feature 2.**
- **Deliverable: A PR for Feature 3 ready for review.**

Week 9 (2nd August to 8th August)

- Make changes to Feature 3 based on the feedback.
- **Deliverable: Completed the implementation of Feature 3.**

Week 10 (9th August to 15th August)

- Final demonstration/tutorial & a blog post.
- **Deliverable: A working Jupyter notebook demonstration and a PR with the first draft in `dask-blog`.**

Week 11 (16th August to 22nd August)

- Final polishing of demonstration/tutorial & a blog post.
- **Deliverable: A published Notebook demonstration & a blog post.**

Completiton Date: 23rd August 2021.

My Commitment

During and after the project, I will help **solve more significant challenges and bugs faced in other Dask projects**, wherever the community needs my help. I'm very enthusiastic about taking up issues (outside my GSoC project) as and when time permits.
I will remain an **active contributor to the Dask Ecosystem**, taking part in discussions for

future projects, implementations, and bugs. There are several miscellaneous non-code tasks that I would like to take up to give back to the community, such as **mentoring and representing the same in tech conferences**.

Why I am suited for this project

I am suited for this project because,

- I have the technical skills required to complete this project: **Python and JavaScript frameworks**, with which **I am experienced enough to tackle any problems** that I might come across.
- Being in my freshman year, **I have a lot of time and enthusiasm to dedicate to the project.**
- I am **willing to learn anything that this project demands**, and I believe within time, I can pick up any skill.
- Being new to the development community, I can **bring a unique and fresh perspective to the project** and Dask.

I have been contributing solely to Dask for the past two weeks, and with the constant support of mentors, I have engaged long enough with the project that **I am clear about the objectives of this project**. After developing well-researched ideas and preparing an actionable plan to achieve them, **I am confident that I can finish the project within the proposed timeline**. I would be thrilled to be a part of Dask this Summer (and even later) to evolve as a developer.

Contributions to `dask/dask*`

- `dask/dask` **[MERGED] PR #7517 for the Issue #1259**
  - I implemented the `dd.DataFrame.product()` (aliased it to call an already existing method `dd.DataFrame.prod()` which serves the same purpose).
  - I learned about the various DataFrame methods from Pandas and how they have been implemented in Dask. **I am confident that I can work**

with large Python code in your codebase, and my understanding of the code structure increases as I spend more time going through Dask repositories.

- `dask/dask-tutorial` **[MERGED] PR #211 for the Issue #206**
    - I updated the Binder Badge Image to the latest badge. It was a simple "Find and Replace" job.
    - I learned about the `dask-tutorial` repository and how it is structured. I also got to know how documentation works at Dask. I had already completed the Dask Tutorial, and after seeing the Jupyter Notebooks, **I am confident that I can contribute to quality documentation.**
- `dask/dask` **[WIP] Issue #1259**
    - I am working on implementing the `dd.DataFrame.axes()` and `dd.DataFrame.duplicated()`.

I will be working on more issues as I get more familiar with Dask's codebase.

About Me

I am **Freyam Mehta**, a Computer Science fresher at the **International Institute of Information Technology, Hyderabad**, India. I have been programming since I was 15. I am passionate about finding clean and innovative solutions to various problems. My current interests include **UI/UX Designing**, **Data Structures and Algorithms**, **Data Science**, **Machine Learning Applications** and, **Open Source Development**.

I like spending time contributing back to society. I also enjoy playing team sports (Cricket, Basketball, and Volleyball), writing, designing, singing, and dancing.

Relevant Programming Experience

I have an **in-depth knowledge** of **C/C++, Python**, and web development technologies such as **HTML, CSS, and JavaScript**. I specifically love **Graphs**! **Graph theory and algorithms** pique my interest a lot. Given a set of nodes and connections, which can abstract anything from city layouts to computer program flow, graph theory can simplify extensive data and help us make sense out of it. I have worked on two Deep Learning (using Python) projects on Image

Classification using Deep Neural Networks:

1. I love cats, so I made a **Cat Classifier** which implements and trains an L-layer Neural Network to classify cats vs. non-cat images by using Logistic Regression and storing a cache to pass information from forward to backpropagation (GitHub Repo).
2. After the success of the Cat Classifier, I tried my hands on a **Number Sign Classifier** which implements and trains a Deep Neural Network in Tensorflow to classify number signs by using Logistic Regression and applying optimization methods such as (Stochastic) Gradient Descent, Momentum, RMSProp, and Adam (GitHub Repo).

I have also worked with **HTML/CSS + Javascript to bring interactivity to static websites**. Specifically, I have experience with **ReactJS** (an amazing JS library that I used for my website and Botomania):

1. Classic games like Tic-Tac-Toe (GitHub Repo) and Number Guessing (GitHub Repo).
2. I have cloned a website from Brittany Chiang. I tweaked around a bit and customized the website to my liking. **I learned so much about JavaScript and React frameworks such as jQuery and Gatsby in this process of playing around with an existing code. I had full control over the output, and I had endless possibilities.** You can find my website https://freyam.netlify.app hosted on Netlify (GitHub Repo).
3. I have worked with my seniors on an interactive game, Botomania (GitHub Repo), that required me to work with React and also work with the back-end (Docker) to communicate data simultaneously.

I love to learn by doing. When I joined Dask, I was confused at the start. So, I thought of using it somewhere myself. Since I was spending so much time on the GSoC organizations list, **I built a simple python scraper (GitHub Repo) using Pandas' DataFrame. This helped me understand the need for the Issue #1259 - To add more Pandas' methods into Dask.**

I have enjoyed immensely working on Dask, and **I continue to be intrigued by the great potential of the new features that could be integrated**, and I cannot wait to be a part of the project.

Open Source Experience

I have been using GitHub for maintaining all my projects for over a year now. I have worked on my college clubs' projects on GitHub, and I am confident about the version control workflow in large codebases.

I have engaged in Open Source institutions previously too. I was intrigued by the concept for a long time and started from my school days. **Open Source contributions have made me realize the importance of a well-documented and easy-to-understand code.** I have participated in 2 editions of Google Code-in (precursor to GSOC) in 2018 and 2019 and have contributed documentation and designs to well known organizations like OpenMRS, LibreHealth, RTEMS and CCExtractor. I have also taken part in the Hacktoberfest 2021 and am an active member of the OSDG@IIIT-H club, and have conducted Open Source Workshops in my college.

Why I want to get selected

Working with Dask would be an excellent opportunity for me. This will be my first time contributing to a robust and complex codebase like yours. Although it is my first time with Dask, **the things I have picked up in the last two weeks have been unimaginable for me.** I have learned about **Parallel workloads using Dask**, **lazy computing**, **Sphinx documentation**, **multiple data visualization libraries/extensions (** `cytoscape` **,** `d3-graphviz` **,** `ipyelk` **, and more)**, **Jupyter** and **JupyterLab**. I am thrilled to take this learning journey even further by working on the project.

I am a curiosity-driven learner. All the programming knowledge I have gained so far results from my endless exploration & self-learning capabilities. **I expect to discover many new things while working in Dask, and I do not hesitate to learn new things to accomplish a goal.** I have been researching different JavaScript libraries and iPython extensions for the project, and this got me excited for the final output of the new visualization tool that I would be developing.

One of the reasons I have been active in Dask is the immensely constructive community, and I will keep myself updated with other developments. The community

was very helpful in getting me started with the development tasks.

More than anything, **I find this project super fun to work with**. This project is inspiring to me because it will give me exposure to Dask's codebase, learn new topics every day and increase my understanding of the inner workings of Python and, specifically, how great visualization can be so crucial.