A. Artifact Appendix

A.1 Abstract

This appendix provides a piece of code and associated data structures used to validate the graph analysis methodology proposed in the paper. The code creates a directed acyclic graph (DAG) consistent with the examples in the paper over networkx and implements several metric computation methods defined in the paper, including the Critical Path, Total Work (TWork), and Modified Critical Path (ModCP) metrics. By running the code, you are able to check the effectiveness of NewLB on different datasets, thus verifying the reproducibility of the paper's conclusions.

A.2 Artifact check-list (meta-information)

- **Algorithm:** Critical path calculation in DAG, TWork (total work) calculation, ModCP (modified critical path) calculation, NewLB calculation, Bottleneck Identification.
- **Program:** Python 3 script using networkx library.
- **Compilation:** Python3 Interpreter
- **Transformations:** N/A.
- **Binary:** N/A
- **Model:** Directed Acyclic Graph (DAG) model representing stages and tasks.
- **Dataset:** Google Cluster, Alibaba Cluster, Azure Cluster
- **Run-time environment:** Python 3.10, or Google Colab
- **Hardware:** No specific hardware requirement; any standard machine running Python 3.
- **Run-time state:** N/A
- **Execution:** N/A
- **Metrics:** NewLB, CPLength, TWork, all three have a unit of seconds.
- **Output:** The computed NewLB of the DAG
- **Experiments:** Running the script to produce the NewLB for the provided dataset.
- **How much disk space required (approximately):** Depends on the size of the dataset, few MB to 100 GB
- **How much time is needed to prepare workflow (approximately):** Depends on the size of the dataset, few minutes to few hours
- **How much time is needed to complete experiments (approximately):** Depends on the size of the dataset, few minutes to few days
- **Publicly available?:** Through GitHub
- **Code licenses (if publicly available)?:** N/A
- **Data licenses (if publicly available)?:** N/A
- **Workflow automation framework used?:** N/A
- **Archived (provide DOI)?:** N/A

A.3 Description

A.3.1 How to access

Please visit our GitHub repository at `https://github.com/JameZ233/cluster-scheduling`

A.3.2 Hardware dependencies

No special hardware dependencies.

A.3.3 Software dependencies

- Python 3.10
- `networkx`
- `matplotlib`
- `graphviz_layout`
- `Pandas`
- `sqlite3`

A.3.4 Datasets

We mainly tested NewLB on two datasets.

- Google Cluster `https://github.com/google/cluster-data`
- Alibaba Cluster `https://github.com/alibaba/clusterdata`
- Azure Cluster `https://github.com/Azure/AzurePublicDataset/tree/master`

For google cluster, we have a script to extract a specific user's task data for evaluation, a user's data is relatively small, and easy for evaluation.

A.3.5 Models

A.4 Installation

No need to install, just clone the whole repository.

A.5 Experiment workflow

1. For analysis of the Google cluster, first you need to extract some samples from the Google cluster, as the data of the whole cluster is extremely big, we provide a jupyter notebook to extract the data from Google database.
2. For analysis of the Alibaba cluster, first you need to download two batch files from the Alibaba Github Page. Then, just run Jupyter Notebook for a thorough analysis.
3. For analysis of the Azure cluster, first you need to download sqlite file '2020 Trace for Packing' from the Azure Github Page. Then just run the python file for the analysis.

A.6 Evaluation and expected results

1. After running google_cluster_analysis.py, you should get the value of NewLb, CPlen, TWork, and the graph after a cut of all of the DAGs. You can also get the visualization of all the graphs, to explicitly see the hierarchy of the DAG.
2. For Aliababa dataset analysis, you should get
   - 'level_graphene_result.txt' and 'resource_utilization_by_job.txt' contains four Graphene parameters for all of the DAGs.
   - 'levels/' directory contains all of the DAG in each level 'instance_level.csv' CSV file temporarily contains all of the instance information of the all of the DAGs in the specific level
3. For Azure dataset analysis, Json files containing four metrics of each DAG will be output

A.7 Experiment customization

1. You can run this code on data of every DAG scheduling cluster for calculating the NewLB. google_cluster_analysis.py is just an example for fitting the Google cluster data into newlb_calculator. All you need to do is convert your data into the format below.
2. For Aliabab analysis, we set default analysis for DAGs with the level of 11. You can change the input file 'level_file' to choose between 0-23, 25, 28, 31, 33, and 53 levels of DAGs to analyze.
3. For Azure analysis, you can change the tenant_id to affect the adjacency list and associated resources for groupings of DAG.

A.8 Notes

Here we introduce the functions in the newlb_calculator.py.

- create_sample_graph() will create a sample DAG that is completely the same as the graph shown in the Graphene paper.
- calc_critical_path() shows the critical path of the input graph which is the longest path length of the input graph.
- calc_twork() calculates the input graph's total work, which is the product of task number, resource and duration.
- cut_dags() is the most complex part of the process of calculate the NewLB, it will properly cut the input DAG into two sub-DAGs. The process of how to doing this is explained in the final report.
- calc_newlb() returns the NewLB of the input DAG, which can be considered as the combination of the above functions.

For Aliabab analysis, only one function is introduced

- load_job_ids(file_path) is a function to load in the chosen level file in 'files/' directory. Input is the path of a '.txt' file

For Azure analysis, we include functions to calculate metrics, as well as helpers to process the data.

- cplen(graph, durations) will take in the graph and calculated durations and produce the cplength of the DAG
- calculate_twork(tasks_resources, resource_capacities) taks in both resources of the DAG to calculate its Total work (Twork)
- ModCP(graph, stages, durations, capacities, duration_resources) takes in both DAG and resources information to calculate the Modified CPLength of the DAG
- construct_stages(graph) takes in the graph of a DAG to calculate and provides the stages within the DAG
- topological_sort(graph) takes in the graph of a DAG to calculate the hierarchial depth for further analysis.