Appendix: Artifact Description/Artifact Evaluation

I. Overview of Contributions and Artifacts

A. Paper's Main Contributions

- C_1 Methods for reshaping high energy physics applications from long-running to near-interactive using TaskVine.
- C_2 Evaluation of serverless task execution for physics application reshaping.
- C_3 Evaluation of utilizing peer transfers to for physics application reshaping.

B. Computational Artifacts

- A_1 github.com/BarrySlyDelgado/SC_TaskVineHEP_AD

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1, C_2, C_3,	Figures 7-8, 11-15 Table 1

II. Artifact Identification

A. Relation To Contributions

The artifact provided, (A_1), includes plotting tools used to generate the original figures in the paper along with the logs used to generate these figures. Additionally, the artifact includes scaled-down experiments used to ensure and verify the reproducibility of the artifact. The provided experiments use a reduced data set to scale down execution time and needed resources. A set of experiments can executed either locally or on distributed resources. The resources provided within the artifact substantiate our paper contributions by providing the original logs of the experiments used in the paper and scaled-down experiments for replication.

B. Expected Results

The scaled-down experiments within the artifact should replicate the significant contributions shown in the paper. Specifically, for figures 7-8 and Figures 11 - 15. As the artifact produces replicated figures for the ones shown in the paper, the expected results are as follows:

Figure 7 - The replication of figure 7 should depict two runs of the application DV3 where the plot on the left is depicting no data transfers between peer workers while the plot on the right will depict more data transfer among peer workers, where the max amount of data transferred between any two peers is smaller when peer transfers are not enabled.

Figure 8 - The replication of figure 8 produces a plot depicting the runtime of tasks during an execution of DV3 in two modes, Tasks and Function Calls. Here, the tasks executed in Function Calls mode should be faster.

Figure 11 - The replication of figure 11 should depict a scaled-down version of the application RS-Triphoton, depicting worker cache usage over the application's runtime. Specifically, this is executed in two modes, single and binary graph reeducation. Binary graph reduction should use less distributed disk space and execute faster.

Figure 12 - The replication of Figure 12 depicts executions of DV3 on various stack configurations as mentioned in the paper. The replicated figure depicts 2 graphs, consisting of pertinent information during runtime. Specifically, concurrent tasks and waiting tasks. This replication contains 3 lines instead of the 4 shown in the paper. The reason being that stack 1 and 2 use different hardware that may not be available for individuals executing artifact experiments. Instead, stacks 1 and 2 are depicted as a single entity. The expected results for this replicated figure should show stack 4 having high concurrent tasks while having low waiting tasks. Stack 3 should have less concurrency resulting in more waiting tasks. The combination of stacks 1 and 2 (noted as stack 2) should have even less concurrency and even more waiting tasks.

Figure 13 - The replication of figure 13 should depict the task stream of executions of DV3 on a set of workers for stacks 3 and 4. Here, the task stream for stack 3 should be more segmented than the stream of stack 4.

Figure 14 - The replication of figure 14 should depict the executions of DV3 and RS-Triphoton at different scales (cores) with workflow executors TaskVine and dask.distributed. Here TaskVine should perform better than dask.distributed at each scale Additionally, scaling is shown for DV3 and RS-TriPhoton.

Figure 15 - The replication for figure 15 should depict the amount of concurrent tasks during an execution of DV3. Here, Concurrency should be maintained high for the majority of the duration of the execution.

C. Estimated Runtime

There are two main experiments presented in the artifact, DV3 and RSTriPhoton. The experiment for DV3 runs in 4 configurations where each configuration is run 3 times. Execution for each configuration is ran at 2 scales. RS-TriPhoton is run in 2 configurations, each 3 times at 3 scales.

1) Artifact Setup: Once downloaded, there is little setup for execution (less than 10 minutes).

2) Artifact Execution: Each execution of DV3 can range from 10 - 15 minutes. For all runs at all scales and configurations, this can take from 4 - 6 hours. Each execution of RS-TriPhoton completes in roughly a minute. This experiment takes roughly 20 minutes.

3) Artifact Analysis: Once execution is complete, the analysis of the reproduced experiments should generate graphs. Graph generation should take less than 10 minutes.

D. Artifact Setup

1) Hardware: If executed on a compute cluster, compute nodes should be equipped with a local disk and have access to a shared filesystem. Additionally, if executed on a a compute cluster, suitable batch schedulers include SLURM, SGE, and HTCondor.

2) Software: The artifact should be executed on a LINUX-based machine. For execution, an installation of Conda $\geq$ 24.4.0 is needed to install software dependencies via a YAML file. Package versions are listed within the YAML file for each experiment's environment.

3) Datasets/Inputs: Datasets are provided within the artifact, where each data set is located under its respective experiment's directory.

4) Installation and deployment: Once Conda is installed and the repository has been downloaded, there are two directories, ”paper_figures” and ”experiments”. First, uncompress large logs by running the command: **”./uncomporess_large_logs.sh”**. Within the experiments directory, traversing to either directory ”DV3” or ”RS-TriPhoton”, individuals can download necessary software dependencies via the command: **”conda env create –name <ENVIRONMENT_NAME> –file=env.yml”**. Then to generate a tarball needed for environment distribution, activate the environment and run the command:**”poncho_package_create $CONDA_PREFIX <dv3-env|rstri-env>.tar.gz”**. Then run the experiment by executing the command: **”./run.sh”**. Once each experiment is complete, a user can run: ”**./gen_graphs.sh**” located in the experiments directory. This will produce the replicated figures mentioned above. Notably, Values are to be entered manually for replications of figure 14. these values can be retrieved from the logs for dask.distributed executions and generated using command: **”vine_plot.py LOG_NAME –compute-time”** for TaskVine executions.

E. Artifact Evaluation

The steps to evaluate the artifact are as follows: once the repo has been downloaded and setup has been completed, execute the experiments located in the directories **experiments/DV3** and **experiments/RS-TriPhoton** as shown in the Artifact Setup (installation and deployment) section above. This produces the logs used for artifact replication. Then generate the replicated figure by executing ”**./gen_graphs.sh**”. For each experiment, 4 core workers are used to reduce the scale. Additionally, 64GB of memory and 100GB of disk space is allocated for execution. Each configuration, for each scale during an experiment is repeated 3 times. Most of these parameters can be modified within the run script. Notably, the size of the datasets for DV3 and RS-TriPhoton are 1.2 and .5 gigabytes respectively. This is to reduce replication compute time.

F. Artifact Analysis

For the figures, information is taken from logs generated on each execution. penitent information is computed from these logs to generate for displays. For each execution, logs will be outputted into the **experiments/logs** directory. To generate the displays the logs are parsed and relevant information is computed. information such as task placement, data transfer events, and task run times are computed from these logs. There are various events that are accumulated to produce a figure. For example, figure 12 is generated via computing task placement and execution duration from the logs.