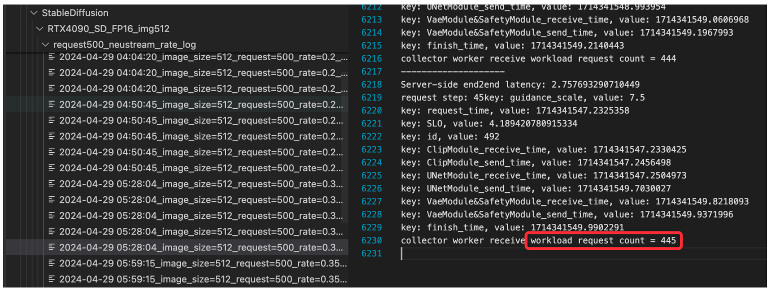
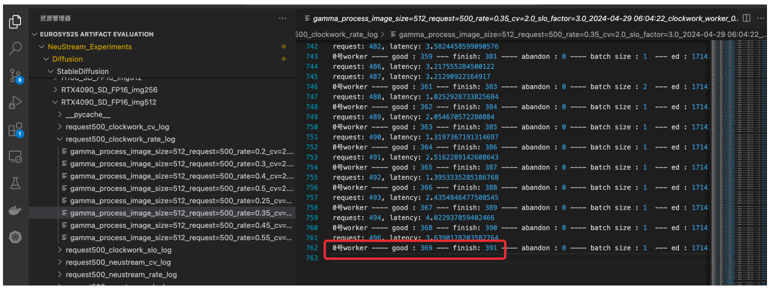
NeuStream Artifact Evaluation

The code repos are shared through google drive. The link is:
If the link is not valid, please check the https://github.com/Fjallraven-hc/NeuStream-AE to see valid link in README.md.

1. Diffusion
   - Models:
     - Stable Diffusion v1.5
       - We test SD on both RTX4090 and H100, with 256x256 & 512x512 image generation.
     - Diffusion Transformer
       - We test origin DiT with its smallest (DiT_S_2) and biggest model (DiT_XL_2) size on 256x256 image generation task.
     - Palette
       - We test Palette on image restoration task on RTX 4090.
   - Experiments
     - The dependency for Stable Diffusion and Diffusion Transformer is in **NeuStream_Experiments/Diffusion/SD_DiT.yaml**, the dependency for Palette is in **NeuStream_Experiments/Diffusion/Palette.yaml.**
     - There are **run_clockwork.sh** and **run_neustream.sh** scripts in each subfolder.
     - The goodput data is extracted from the serving log, you can see **NeuStream_Experiments/Diffusion/plot_high_goodput.py** and **NeuStream_Experiments/Diffusion/plot_low_goodput.py** for the data in Figure 11 and Figure 12. The data for plot locates at the last line of the serving log in each subfolder, like below.

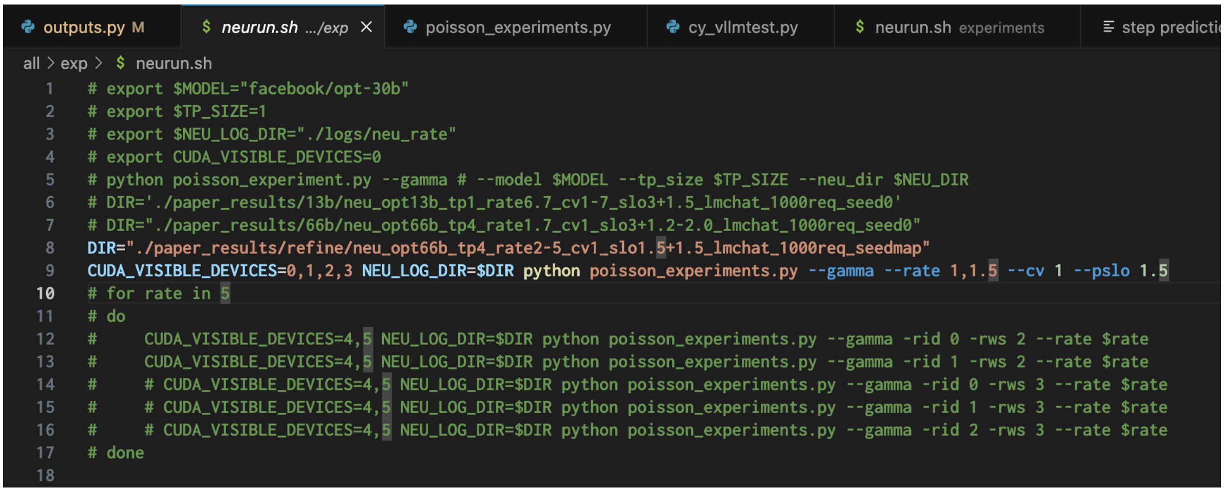




2. LLM
   - We use the OPT model family to test the workload. For convenience, we didn't include origin model parameters in the Artifacts. Because under our evaluation in Figure 14 & 15, we choose the co-locate settings, so the prefill and decode instance are located in same

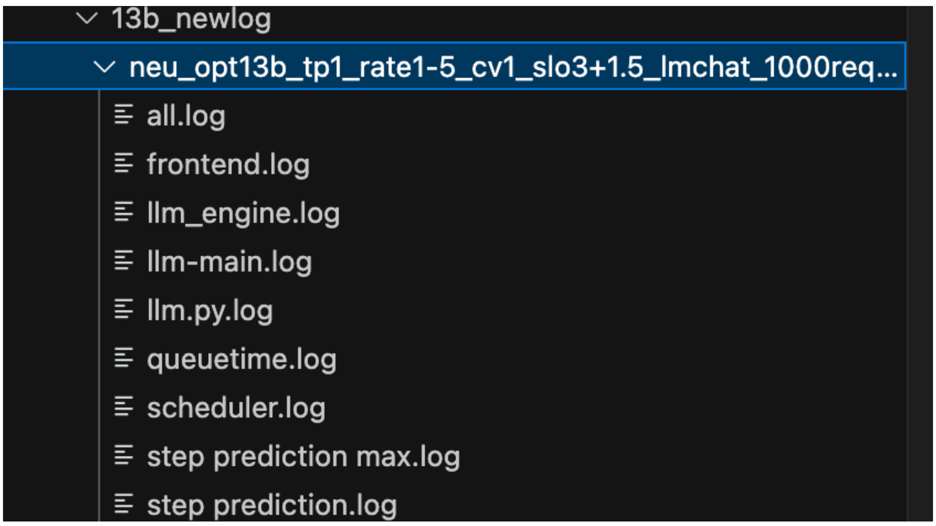
process, and the implementation transforms to the execution order of prefilling and decoding, as shown in paper's Figure 10.

- To reproduce the data, it needs the NVIDIA A6000 and H100 accelerators.
- In the OPT–LLM folder, you can see three subfolders, **experiments** is for experiments, **neusim** is used for simulating the latency of prefill & decode execution when under different batch sizes or prefix lengths, and **vllm_files** is the modified vLLM to support NeuStream in co–locating setting. To setup the environment, first create the conda env through the **NeuStream_Experiments/LLM–OPT/experments/conda.yaml**, and the install our modified vLLM backend in **NeuStream_Experiments/LLM–OPT/vllm_files/v0.5.4.tar.gz**.
- **Experiments**:
  - To launch NeuStream, run **NeuStream_Experiments/LLM–OPT/experments/neurun.sh**
  - To launch origin vLLM, run **NeuStream_Experiments/LLM–OPT/experments/vllmrun.sh**
  - You can change the rate, cv, or slo through modifying the scripts, like below picture

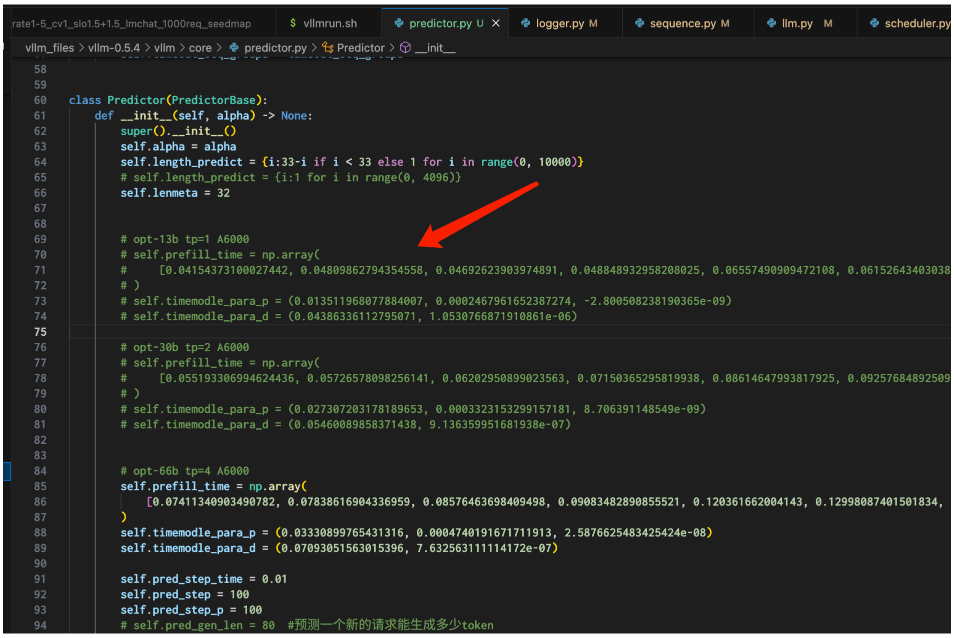



```
# export $MODEL="facebook/opt-30b"
# export $TP_SIZE=1
# export $NEU_LOG_DIR="./logs/neu_rate"
# export CUDA_VISIBLE_DEVICES=0
# python poisson_experiment.py --gamma # --model $MODEL --tp_size $TP_SIZE --neu_dir $NEU_DIR
# DIR='./paper_results/13b/neu_opt13b_tp1_rate6.7_cv1-7_slo3+1.5_lmchat_1000req_seed0'
# DIR="./paper_results/66b/neu_opt66b_tp4_rate1.7_cv1_slo3+1.2-2.0_lmchat_1000req_seed0"
DIR="./paper_results/refine/neu_opt66b_tp4_rate2-5_cv1_slo1.5+1.5_lmchat_1000req_seedmap"
CUDA_VISIBLE_DEVICES=0,1,2,3 NEU_LOG_DIR=$DIR python poisson_experiments.py --gamma --rate 1,1.5 --cv 1 --pslo 1.5
# for rate in 5
# do
#     CUDA_VISIBLE_DEVICES=4,5 NEU_LOG_DIR=$DIR python poisson_experiments.py --gamma -rid 0 -rws 2 --rate $rate
#     CUDA_VISIBLE_DEVICES=4,5 NEU_LOG_DIR=$DIR python poisson_experiments.py --gamma -rid 1 -rws 2 --rate $rate
#     # CUDA_VISIBLE_DEVICES=4,5 NEU_LOG_DIR=$DIR python poisson_experiments.py --gamma -rid 0 -rws 3 --rate $rate
#     # CUDA_VISIBLE_DEVICES=4,5 NEU_LOG_DIR=$DIR python poisson_experiments.py --gamma -rid 1 -rws 3 --rate $rate
#     # CUDA_VISIBLE_DEVICES=4,5 NEU_LOG_DIR=$DIR python poisson_experiments.py --gamma -rid 2 -rws 3 --rate $rate
# done
```

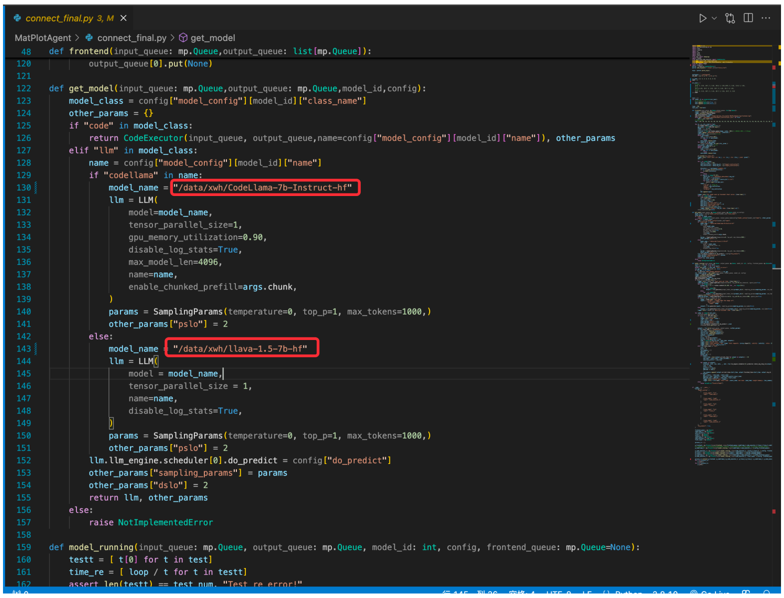
  - The above command support multiple parameters in rate, cv or slo, but we recommend only passing one group multiple parameters.
  - Model size and corresponding tensor parallelism need mannually specification. Our search shows that the best tensor parallelism for OPT 13B, 30B, and 66B is 1, 2, and 4 respectively. NUM DEDUP is the number of repeated experiment groups in a setting, which can be set to 1 during the test. The final output log contains a step prediction max.log, which will output the highest goodput value in each of these experiments separately.



- Besides, when test different models, user has to manually specify the prediction/simulation data for the certain model.



3. Multi-Agent
- We use the MatPlotAgent(https://github.com/thunlp/MatPlotAgent) to test the workload. The environment need NVIDIA RTX4090 & H100 to reproduces the data in paper.
- Firstly, two models' parameter need to be downloaded.
  - https://huggingface.co/llava-hf/llava-1.5-7b-hf
  - https://huggingface.co/meta-llama/CodeLlama-7b-Instruct-hf



- Then, modify the model path in **MatPlotAgent/connect_final.py**, as shown below.
- Secondly, see README.md in MatPlotAgent for experiments detail.