Benchmarking Inference Serving of Llama 2

Llama2 Inferencing Benchmarking Tool

Llama2 Inferencing Benchmarking Tool

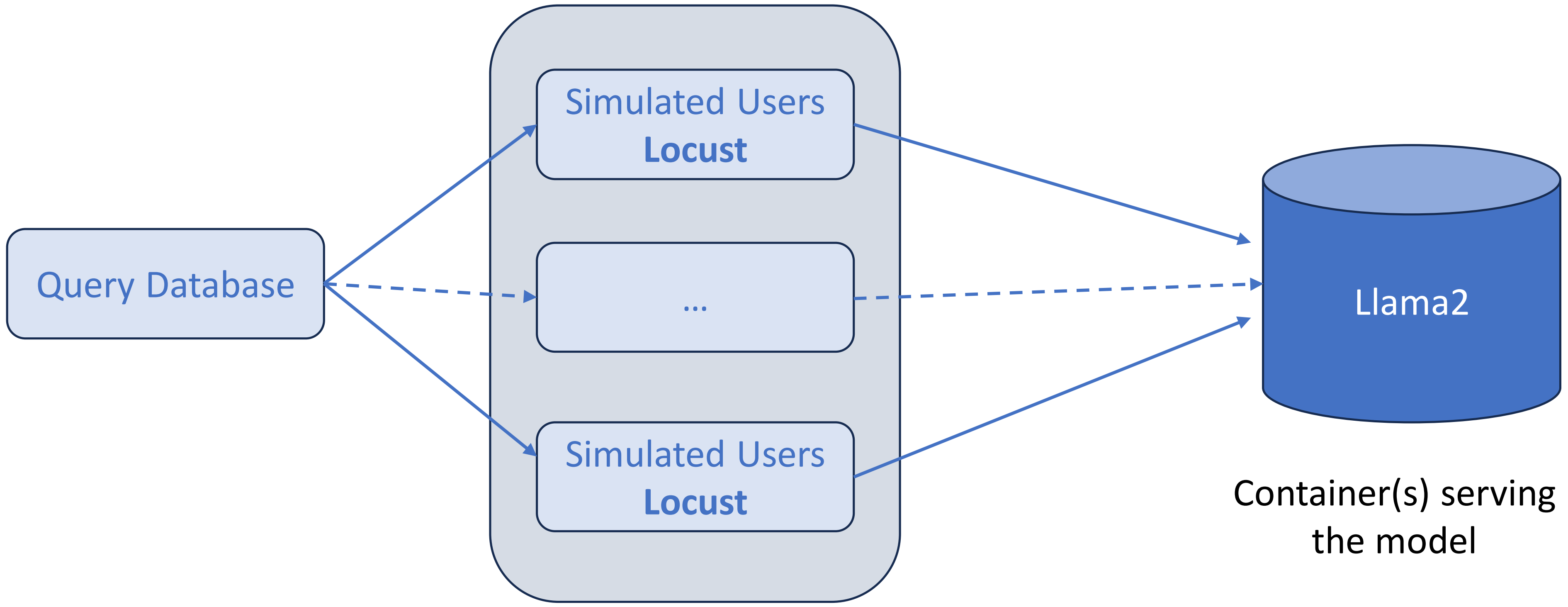
Goal of the Inferencing benchmark tool is to identify

- Latency of each request made and measured in millisecond/token
- Latency of the Time Taken for the First Token (TTFT) – higher the latency drastically affects user experience
- Throughput measured in number of tokens/second

This is measured with varying sized of

- Input Tokens (query length)
- Output Tokens (response length)
- Simulating parallel users

Llama2 Inferencing Benchmarking Tool



Currently user simulated from a single node

Container(s) serving the model

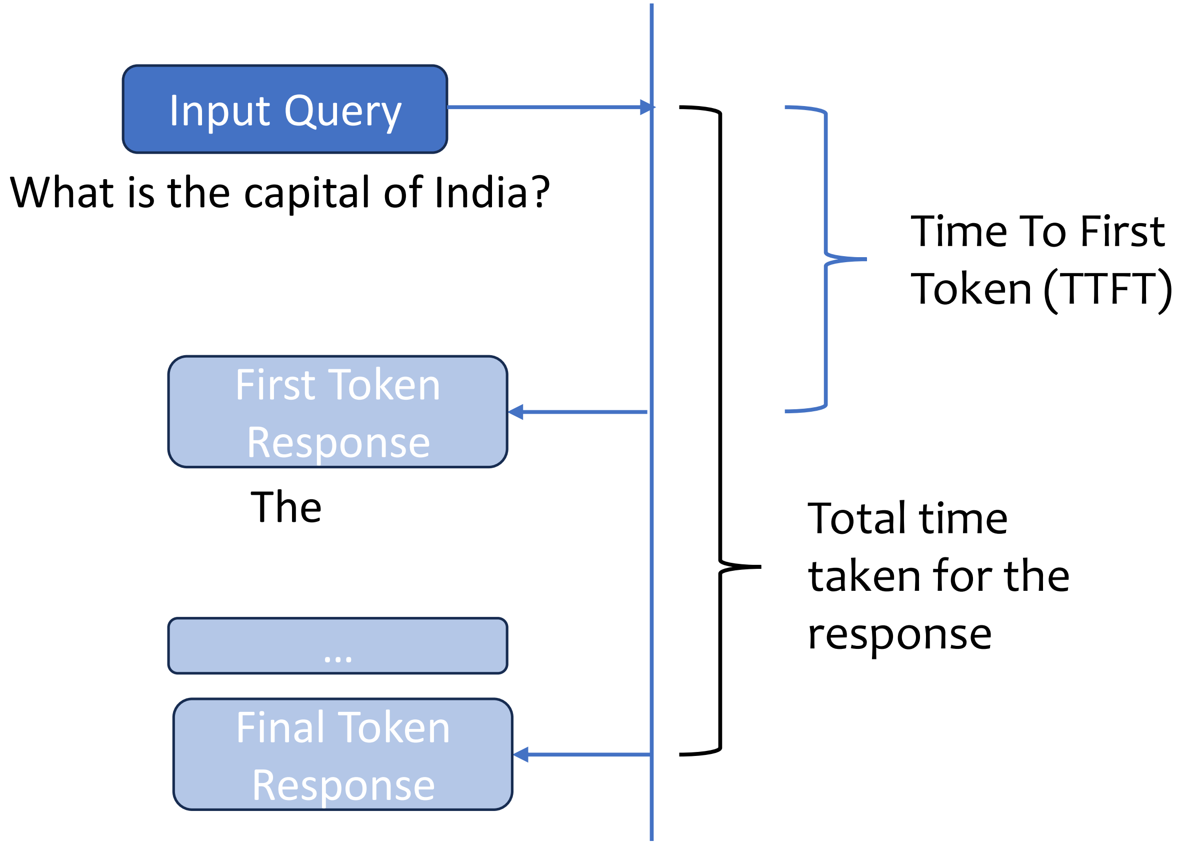
Latency and Throughput

Latency

- Latency is measured total time taken to output all the tokens
- Latency = Total Time taken for the response / number of tokens
- Latency is generally in milliseconds

Throughput

- This is a measure of total number of tokens per second
- Here both Input tokens and Output tokens are considered


Input Query
What is the capital of India?
First Token Response
The
...
Final Token Response
Time To First Token (TTFT)
Total time taken for the response

Running the Benchmark

Generating Input Dataset

Dataset : “pvduy/sharegpt_alpaca_oa_vicuna_format”

Steps involved in dataset generation

- This dataset consists of nearly 324k prompts.
- Using LLaMA tokenizer prompts are tokenized.
- Input tokens are of sizes 32, 64, 128, 256, 512, 1024(1k), 2048(2k) each with 1000 queries are saved in 7 different .CSV files.

Inference Testing Plan – Identify ideal configuration (core/memory) for different models

Benchmark Input Query

generate_query_db.py

Process Hugging Face Query Dataset to generate

- 7 files with 1000 queries each for input token 32, 64, 128, 256, 512, 1k, 2k
- Input token size is not exact and is usually plus/minus 10 tokens the expected input token, example, file with 128 input tokens has queries with input token size 118 – 138

profile_container.py

Start monitoring / profiling the container
stats (CPU/Memory) during the benchmark .

Benchmark Script

sut_loop_wrapper.py

- Start the container with different cores/memory running the model
- Obtain the model endpoint
- Run LLM_inference_Benchmark.py
- Stop the container

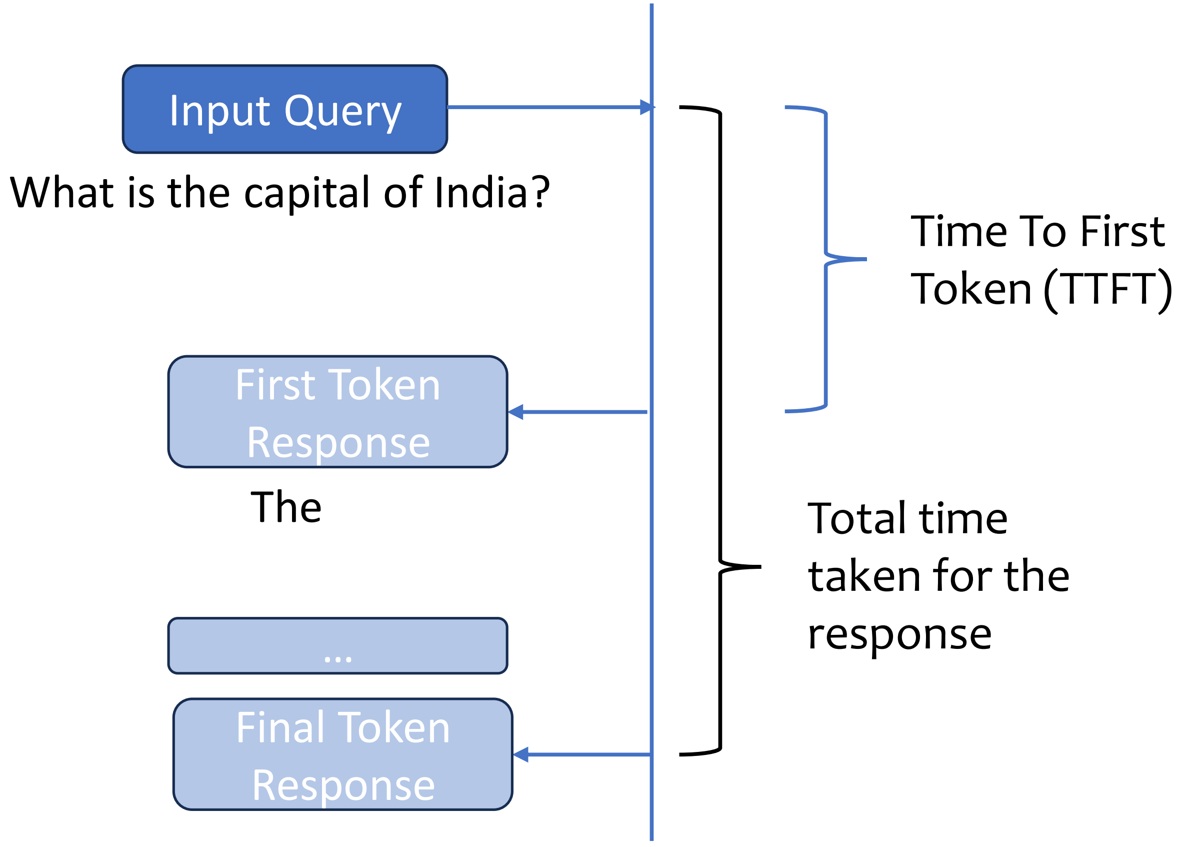
→

llm_inference_benchmark.py

- llm_inference_master.py: Run Inference benchmark on the llm
  a) for increasing parallel users ,
  b) run the benchmark for different combinations of Input tokens, Output tokens to obtain (latency, performance, TTFT) and
  c) write output in a different directory to csv
- llm_result_analysis.py: Analyze all different output files for each combination to
  - Chart TTFT/Latency/Performance
  - Identify peak performance
  - Chart CPU/Memory for each user test
  - Chart TTFT/Latency/Performance

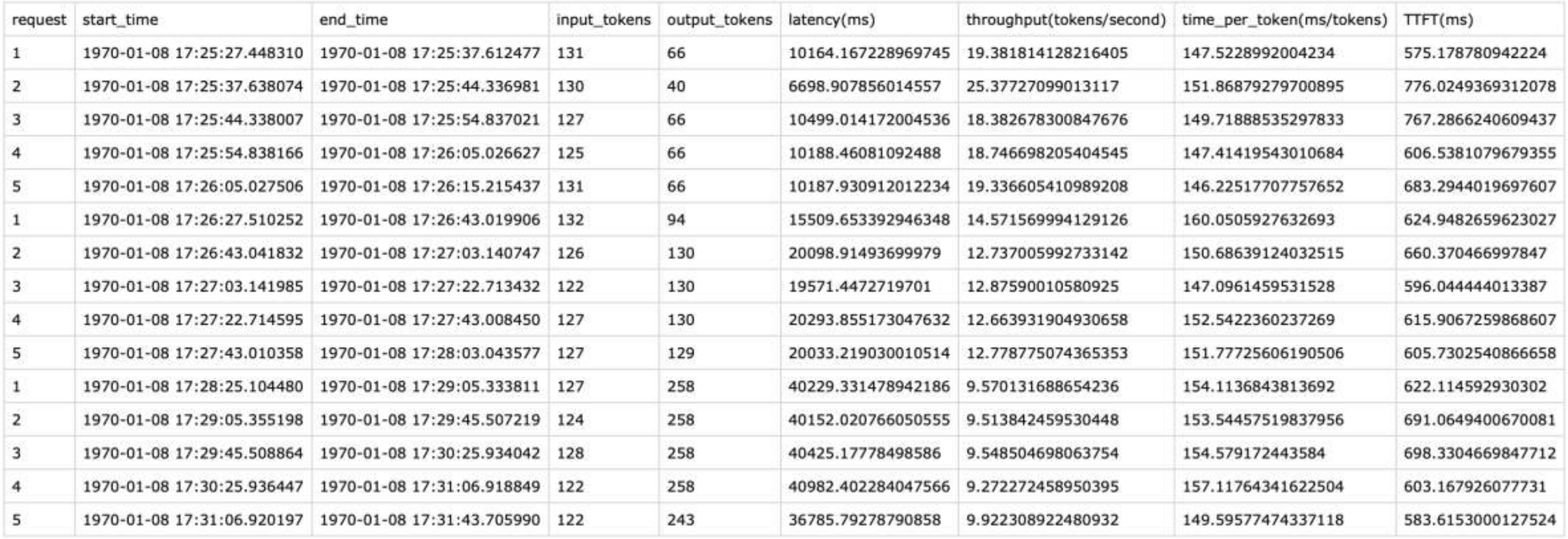
Formula Used

- TTFT – Time for First Token
- Latency / token – Time taken for all tokens excluding the first
- Throughput - Tokens / second – Total number of tokens / total time


Input Query
What is the capital of India?
First Token Response
The
...
Final Token Response
Time To First Token (TTFT)
Total time taken for the response

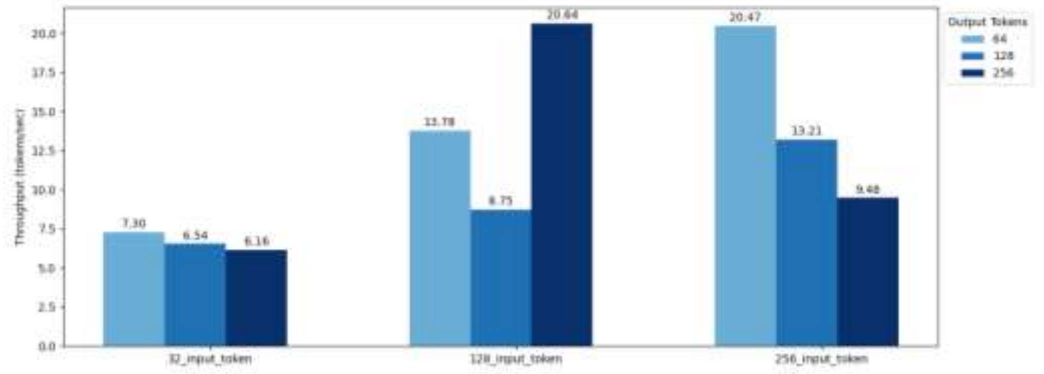
Sample Output – 1 User
Input tokens 128 – Output tokens 64, 128, 256

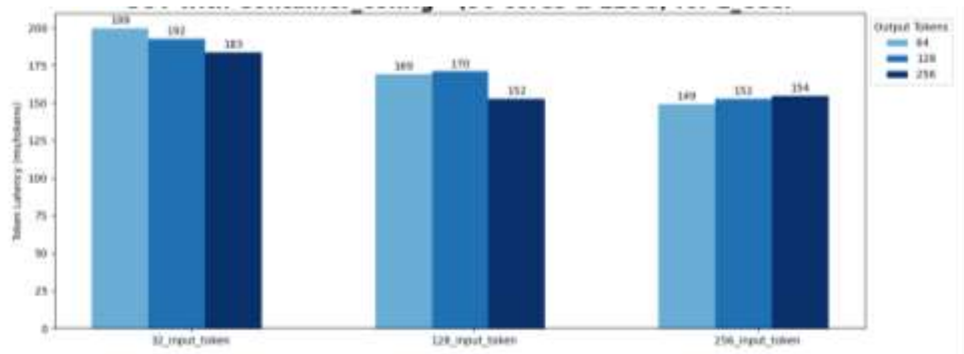
Latency per token = (latency – TTFT) / (output tokens -1)

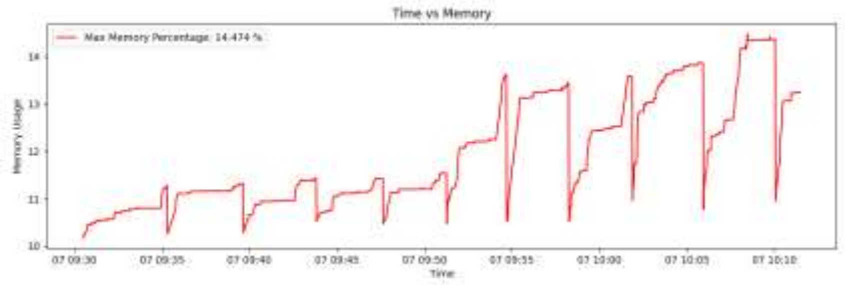


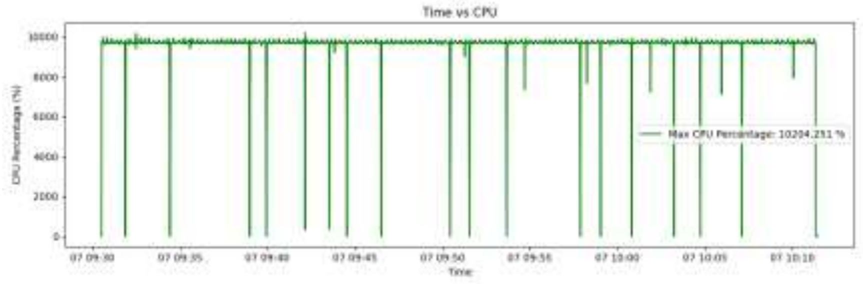
request	start_time	end_time	input_tokens	output_tokens	latency(ms)	throughput(tokens/second)	time_per_token(ms/tokens)	TTFT(ms)
1	1970-01-08 17:25:27.448310	1970-01-08 17:25:37.612477	131	66	10164.167228969745	19.381814128216405	147.5228992004234	575.178780942224
2	1970-01-08 17:25:37.638074	1970-01-08 17:25:44.336981	130	40	6698.907856014557	25.37727099013117	151.86879279700895	776.0249369312078
3	1970-01-08 17:25:44.338007	1970-01-08 17:25:54.837021	127	66	10499.014172004536	18.382678300847676	149.71888535297833	767.2866240609437
4	1970-01-08 17:25:54.838166	1970-01-08 17:26:05.026627	125	66	10188.46081092488	18.746698205404545	147.41419543010684	606.5381079679355
5	1970-01-08 17:26:05.027506	1970-01-08 17:26:15.215437	131	66	10187.930912012234	19.336605410989208	146.22517707757652	683.2944019697607
1	1970-01-08 17:26:27.510252	1970-01-08 17:26:43.019906	132	94	15509.653392946348	14.571569994129126	160.0505927632693	624.9482659623027
2	1970-01-08 17:26:43.041832	1970-01-08 17:27:03.140747	126	130	20098.91493699979	12.737005992733142	150.68639124032515	660.370466997847
3	1970-01-08 17:27:03.141985	1970-01-08 17:27:22.713432	122	130	19571.4472719701	12.87590010580925	147.0961459531528	596.044444013387
4	1970-01-08 17:27:22.714595	1970-01-08 17:27:43.008450	127	130	20293.855173047632	12.663931904930658	152.5422360237269	615.9067259868607
5	1970-01-08 17:27:43.010358	1970-01-08 17:28:03.043577	127	129	20033.219030010514	12.778775074365353	151.77725606190506	605.7302540866658
1	1970-01-08 17:28:25.104480	1970-01-08 17:29:05.333811	127	258	40229.331478942186	9.570131688654236	154.1136843813692	622.114592930302
2	1970-01-08 17:29:05.355198	1970-01-08 17:29:45.507219	124	258	40152.020766050555	9.513842459530448	153.54457519837956	691.0649400670081
3	1970-01-08 17:29:45.508864	1970-01-08 17:30:25.934042	128	258	40425.17778498586	9.548504698063754	154.579172443584	698.3304669847712
4	1970-01-08 17:30:25.936447	1970-01-08 17:31:06.918849	122	258	40982.402284047566	9.272272458950395	157.11764341622504	603.167926077731
5	1970-01-08 17:31:06.920197	1970-01-08 17:31:43.705990	122	243	36785.79278790858	9.922308922480932	149.59577474337118	583.6153000127524

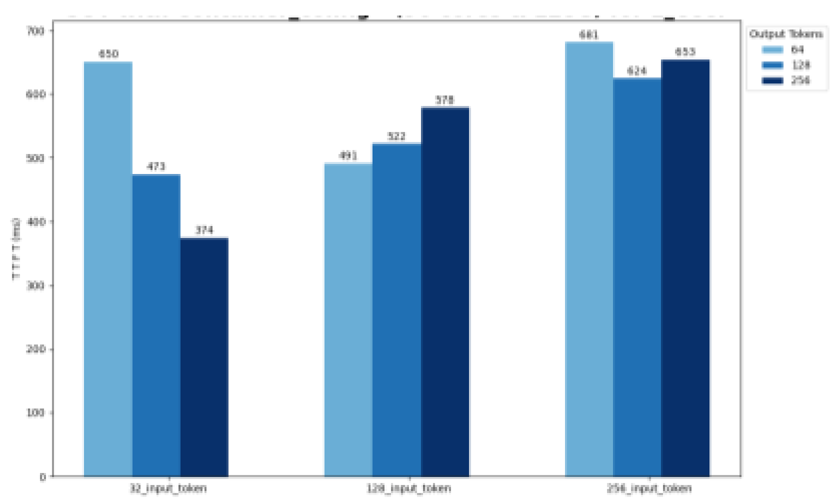
Sample Output











Thank You!