EE508 Project - Phase_2

Asmita Mohanty
Shaun Almeida

Files need to be changed to disable KV Caching:

1. models.py
2. generation.py
3. inference.py

All changes to each file are highlighted in the red box.
Note: There are few masked comments in green in the code which are the debug prints done to understand the differences in the output results between with vs without KV Caching. They are not added to the final code. Debug print outputs shared at the end of this file.

1. models.py

Without KV Caching:

- Keys & Values of the *entire* token per sequence is computed
- Doesn't require to store the keys & values in cache anymore

With KV Caching:

- Keys & Values of the *current* token per sequence is only computed
- Stores the computed results in the cache & later assigns the entire cache that consists of the past tokens as well as the current tokens to the keys & values for downstream computation

```
if self.kv_caching:
    self.cache_k = self.cache_k.to(xq)
    self.cache_v = self.cache_v.to(xq)

    self.cache_k[:bsz, start_pos : start_pos + seqlen] = xk
    self.cache_v[:bsz, start_pos : start_pos + seqlen] = xv

    keys = self.cache_k[:bsz, : start_pos + seqlen]
    values = self.cache_v[:bsz, : start_pos + seqlen]

else:
    keys = xk
    values = xv
```

2. **generation.py**

   - Tokens per sequence are passed as batched tensors

Without KV Caching:

- Pass the entire token per sequence to compute the logits

With KV Caching:

- Pass only the current token per sequence to compute the logits

```
if kv_caching:
    #for t,seq in enumerate(tokens[:, prev_pos:cur_pos]):
    #   print(f"W/ KV Cache, Tokens: {t}, seq:{seq.tolist()} of length:", len(seq.tolist()),
    # " decoded tokens:", tokenizer.decode(seq.tolist()))
    logits = self(tokens[:, prev_pos:cur_pos], prev_pos)
else:
    #for t,seq in enumerate(tokens[:, :cur_pos]):
    #  print(f"W/o KV Cache, Tokens: {t}, seq:{seq.tolist()} of length:", len(seq.tolist()),
    # " decoded tokens:", tokenizer.decode(seq.tolist()))
    logits = self(tokens[:, :cur_pos], prev_pos)
```

3. **Inference.py**

- Toggle the kv caching boolean flag

```
model.eval()
results = model.generate(tokenizer, prompts, max_gen_len=64, temperature=0.6, top_p=0.9, kv_caching=False, device=device)
```

In benchmark_inference.py: Toggle the batch_size & kv_caching flag to compare the KV caching results with different batch sizes.

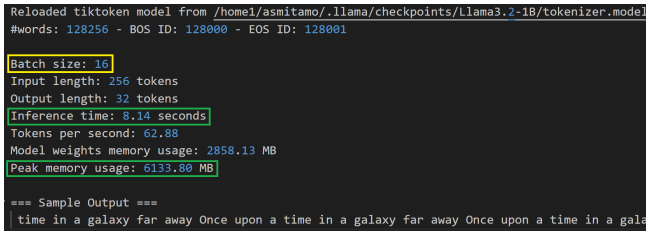
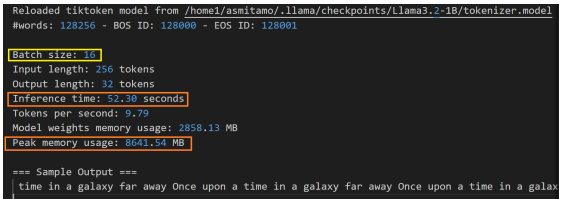
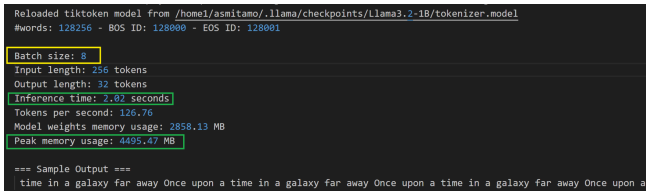
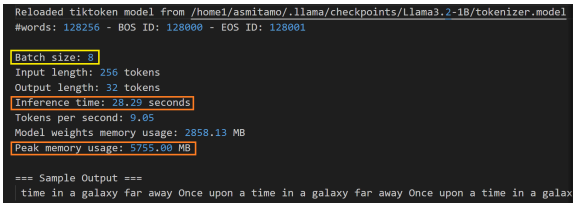
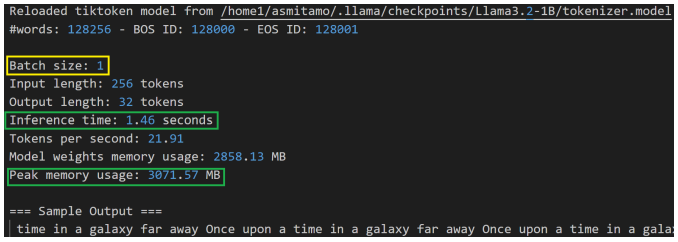
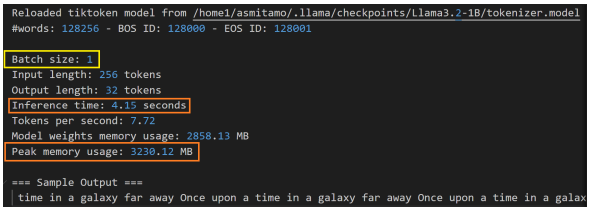
```
if __name__ == "__main__":
    benchmark_inference(batch_size=1, input_len=256, output_len=64, kv_caching=False)
```

Output Results:

All outputs tested with input length = 256 tokens & output length = 32 tokens

KV Caching	Batch size = 1	Batch size = 8	Batch size = 16
With KV Cache	Peak Memory: **3071.57MB** Run Time: **1.46s**	Peak Memory: **4495.47MB** Run Time: **2.02s**	Peak Memory: **6133.8MB** Run Time: **8.14s**
Without KV Cache	Peak Memory: **3230.12MB** Run Time: **4.15s**	Peak Memory: **5755MB** Run Time: **28.29s**	Peak Memory: **8641.54MB** Run Time: **52.3s**

Output Logs:

Batch Size	With KV Cache	Without KV Cache
16		
8		
1		

Additional Data

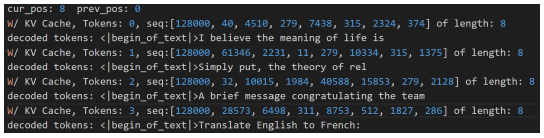
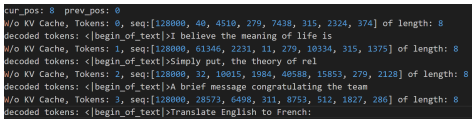
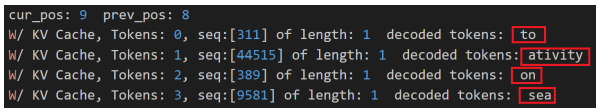
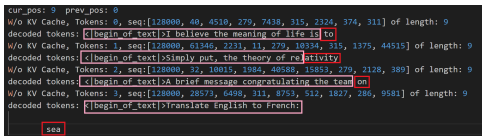
Debug Outputs explaining with/without KV Caching differences

1. Given prompts (same for both w/ KV vs w/o KV cache):

```
Prompts: ['I believe the meaning of life is', 'Simply put, the theory of relativity states that ', 'A brief message congratul
Prompt tokens: [[128000, 40, 4510, 279, 7438, 315, 2324, 374], [128000, 61346, 2231, 11, 279, 10334, 315, 1375, 44515, 5415, 
length of token: 8  decoded tokens: <|begin_of_text|>I believe the meaning of life is
length of token: 12  decoded tokens: <|begin_of_text|>Simply put, the theory of relativity states that
length of token: 21  decoded tokens: <|begin_of_text|>A brief message congratulating the team on the launch:

        Hi everyone,

        I just
length of token: 41  decoded tokens: <|begin_of_text|>Translate English to French:

        sea otter => loutre de mer
        peppermint => menthe poivrée
        plush girafe => girafe peluche
        cheese =>
```

2. During token generation, we iterate over the batched tensor which contains sequences of varying lengths. Starting the 2nd iteration, inference “with KV cache” will generate only 1 token while inference “without KV cache” will output the newly generated token along with the previous/past tokens. Debug outputs shown for first 3 iterations.

Legend:
Pink - Past/Previous tokens
Red - Newly generated tokens

Iteration #	With KV Cache	Without KV Cache
1		
2		
3