

Bechmark

Benchmark for LLM engines

bench.py

- [ollama ??????? vllm ??????????????_ollama vllm-CSDN??](#)
- YT: [ollama vs vllm - ???????? ollama ? vllm ??????? - YouTube](#)

```
import aiohttp
import asyncio
import time
from tqdm import tqdm

import random

questions = [
    "Why is the sky blue?", "Why do we dream?", "Why is the ocean salty?", "Why do leaves change color?",
    "Why do birds sing?", "Why do we have seasons?", "Why do stars twinkle?", "Why do we yawn?",
    "Why is the sun hot?", "Why do cats purr?", "Why do dogs bark?", "Why do fish swim?",
    "Why do we have fingerprints?", "Why do we sneeze?", "Why do we have eyebrows?", "Why do we have hair?",
    "Why do we have nails?", "Why do we have teeth?", "Why do we have bones?", "Why do we have muscles?",
    "Why do we have blood?", "Why do we have a heart?", "Why do we have lungs?", "Why do we have a brain?",
    "Why do we have skin?", "Why do we have ears?", "Why do we have eyes?", "Why do we have a nose?",
    "Why do we have a mouth?", "Why do we have a tongue?", "Why do we have a stomach?", "Why do we have intestines?",
    "Why do we have a liver?", "Why do we have kidneys?", "Why do we have a bladder?", "Why do we have a pancreas?",
    "Why do we have a spleen?", "Why do we have a gallbladder?", "Why do we have a thyroid?", "Why do we have adrenal glands?",
    "Why do we have a pituitary gland?", "Why do we have a hypothalamus?", "Why do we have a thymus?",
    "Why do we have lymph nodes?",
    "Why do we have a spinal cord?", "Why do we have nerves?", "Why do we have a circulatory system?", "Why do we have a respiratory system?",
    "Why do we have a digestive system?", "Why do we have an immune system?"
```

```
]
```

```
async def fetch(session, url):
    """
    Args:
        session (aiohttp.ClientSession): aiohttp session
        url (str): URL

    Returns:
        tuple: (tokens, request_time)
    """
    start_time = time.time()

    # Choose a random question
    question = random.choice(questions) # <--- random question

    # Format the question
    # question = questions[0] # <--- first question

    # Create the payload
    json_payload = {
        "model": "llama3:8b-instruct-fp16",
        "messages": [{"role": "user", "content": question}],
        "stream": False,
        "temperature": 0.7 # 0.7 temperature
    }

    async with session.post(url, json=json_payload) as response:
        response_json = await response.json()
        end_time = time.time()
        request_time = end_time - start_time
        completion_tokens = response_json['usage']['completion_tokens'] # completion tokens
        return completion_tokens, request_time

async def bound_fetch(sem, session, url, pbar):
    # Acquire semaphore
    async with sem:
        # Fetch
        result = await fetch(session, url)
        pbar.update(1)
        return result
```

```

async def run(load_url, max_concurrent_requests, total_requests):
    """
    """

    semaphore = asyncio.Semaphore(max_concurrent_requests)

    async def fetch(url):
        async with semaphore:
            async with aiohttp.ClientSession() as session:
                async with session.get(url) as response:
                    return response.status, response.time

    tasks = []

    for _ in range(total_requests):
        task = asyncio.ensure_future(fetch(load_url))
        tasks.append(task)

    results = await asyncio.gather(*tasks)

    completion_tokens = sum(result[0] for result in results)

    response_times = [result[1] for result in results]

    return completion_tokens, response_times

if __name__ == '__main__':

```

```

import sys

if len(sys.argv) != 3:
    print("Usage: python bench.py <C> <N>")
    sys.exit(1)

C = int(sys.argv[1]) # [ ] [ ] [ ] [ ]
N = int(sys.argv[2]) # [ ] [ ] [ ]

# vllm [ ] ollama [ ] openai [ ] api [ ] [ ] [ ] [ ] [ ] [ ]
url = 'http://localhost:11434/v1/chat/completions'

start_time = time.time()
completion_tokens, response_times = asyncio.run(run(url, C, N))
end_time = time.time()

# [ ] [ ] [ ] [ ]
total_time = end_time - start_time
# [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
avg_time_per_request = sum(response_times) / len(response_times)
# [ ] [ ] [ ] [ ] token [ ] [ ]
tokens_per_second = completion_tokens / total_time

print(f'Performance Results:')
print(f' Total requests      : {N}')
print(f' Max concurrent requests : {C}')
print(f' Total time             : {total_time:.2f} seconds')
print(f' Average time per request : {avg_time_per_request:.2f} seconds')
print(f' Tokens per second      : {tokens_per_second:.2f}')

```

Revision #3

Created 22 July 2024 19:51:19 by Admin

Updated 11 November 2024 09:53:52 by Admin