

Llama 3.1 Model Deployment Research Report

Lauren Mok and Nick Hao

August 16, 2024

ABSTRACT

This study addresses the critical gap in understanding the specific hardware configurations necessary to efficiently operate large language models, with a focus on the GPU resources needed for the LLaMa series. We conducted a series of performance tests on two variants, LLaMa3.1-8b and LLaMa3.1-70b, to evaluate how different input/output token quantities and concurrent request loads impact model performance. Utilizing Microsoft Azure's scalable cloud computing resources, we explored various GPU setups, including single and dual NVIDIA A100 GPU configurations. Our research provides empirical data on the models' accuracy, coherence, and computational efficiency, offering insights into the optimal number of GPUs and memory requirements. The findings contribute to a set of practical guidelines for the deployment and scaling of LLaMa3.1 models, facilitating their broader adoption and efficient use. This report aids researchers and industry practitioners in making informed decisions about resource allocation, potentially reducing costs and improving the implementation of these powerful AI tools.

Keywords

Large Language Models, LLaMa3.1, GPU Configuration, Computational Resources, Performance Testing, Concurrent Requests, AI Deployment, Cloud Computing, NVIDIA A100, Model Scalability

Table of Contents

INTRODUCTION	3
PRELIMINARY RESEARCH	3
METHODOLOGY	4
ENVIRONMENT SETUP	4
PERFORMANCE TESTING	4
STRUCTURED QUESTION TESTING.....	5
RESULTS	5
OUTPUT TOKEN TEST	5
INPUT TOKEN TEST	8
SYSTEM AND GPU MEMORY UTILIZATION	8
STRUCTURED QUESTION TESTING.....	10
DISCUSSION	10
CONCLUSION	11
Appendix A: Ollama and GPU Specifications	12
Appendix B: Running LLaMa 3.1 Locally with Ollama.....	13
Appendix C: Python Scripts for Testing Model Performance.....	15
Appendix E: Tools for Monitoring Resource Usage	17
Appendix F: Performance Testing - Individual Response Times	18
Appendix G: Questions Used to Test Model Performance.....	20

INTRODUCTION

In recent years, artificial intelligence has made significant strides with large language models (LLMs) like OpenAI's GPT-3 and Meta's LLaMa. The LLaMa series, particularly the latest LLaMa3.1 with 70 billion parameters, stands out for its performance and scalability. However, the increasing complexity of these models demands a thorough understanding of the hardware required to run them efficiently.

Despite their impressive capabilities, there is a notable gap in knowledge regarding the specific GPU resources needed for models like LLaMa3.1-70b. This gap poses challenges for researchers and practitioners aiming to deploy these models effectively. Current user guides provide some insights into the hardware needed, but comprehensive, empirically-backed guidelines are lacking. Users often resort to trial and error or rely on anecdotal evidence, which can lead to suboptimal resource allocation and increased costs. The lack of clear, detailed information on GPU requirements for running LLaMa3.1-70b models hinders the broader adoption and efficient utilization of these powerful tools. Therefore, it is crucial to systematically investigate and document the GPU resources necessary to run the LLaMa3.1-70b model.

This study aims to explore the specific GPU configurations that can support the LLaMa3.1-70b model, including the number and type of GPUs, memory requirements, and computational capabilities. Additionally, the research seeks to provide practical guidelines for optimizing GPU usage to ensure efficient and cost-effective deployment of the LLaMa3.1-70b model. The findings will help both researchers and industry practitioners make informed decisions about resource allocation.

PRELIMINARY RESEARCH

Our initial investigation into the deployment of advanced large language models like LLaMa3.1 revealed a notable lack of clear, empirical guidelines on hardware requirements. While resources such as user guides and tutorials offer some direction, they rarely delve into the specifics necessary for optimal configuration. For example, existing tutorials, including a notable one on YouTube, demonstrate deploying the LLaMa3.1-70b model with dual 40GB A100 GPUs but fail to compare this with using a single 80GB A100 GPU. This omission leaves users without clear guidance on the trade-offs between these configurations, particularly in terms of performance and cost.

Moreover, there is a scarcity of information regarding the handling of concurrent requests. Users may be left to rely on trial and error, which can lead to inefficient resource use and increased operational costs. The preliminary research thus underlines the need for a study that provides definitive insights into not only the GPU configurations that best support models like LLaMa3.1-70b but also how these configurations handle varying levels of concurrency. By addressing these gaps, our research aims to offer actionable information that can lead to more informed and efficient deployment of large language models.

METHODOLOGY

ENVIRONMENT SETUP

This study aims to assess the performance and resource demands of two variants of the LLaMa model, specifically LLaMa3.1-8b and LLaMa3.1-70b. Our objective was to conduct performance tests on both models while monitoring system and GPU resource usage. We designed tests to understand how varying input/output token quantities and the number of concurrent requests affect these models' performance. Key performance metrics, such as accuracy, coherence, and computational efficiency, were the focus.

To this end, we manipulated the number of tokens provided as input and required as output by the models. Additionally, we tested the models' ability to handle multiple simultaneous interactions by varying the number of concurrent requests. These experiments were conducted using Microsoft Azure's scalable cloud computing resources. We chose specific GPU-enabled instances for each model to provide the necessary computational power.

For the 8b model, we used an instance equipped with the NVIDIA T4 GPU, whereas testing for the 70b model was split into two phases. The first phase involved an instance with a single NVIDIA A100 GPU, and for the second phase we escalated the setup to include two A100 GPUs. Appendix A contains the specifications for both GPUs. To facilitate running the models, we utilized Ollama, an open-source tool that enables the operation of large language models on local machines. Instructions for running the models with Ollama and setting up the compute instances on Azure are provided in appendix B. The virtual machine configurations used were as follows:

Virtual Machine Type	Cores	RAM	Disk Size
Standard_NC4as_T4_v3	4	28 GB	176 GB
Standard_NC24ads_A100_v4	24	220 GB	64 GB

Table 1: "Specifications of Virtual Machine Types"

PERFORMANCE TESTING

To evaluate the models, we developed two Python scripts. The first script explores how model performance responds to variations in output token count and concurrent request numbers. It instructs the model to create a story with a predefined number of words. The second tests the impact of concurrent request numbers on model performance while keeping input token count constant, directing the model to generate a title of a specified word count for a given text. We named the two tests *output token test* and *input token test* respectively.

The execution of these scripts provides two primary performance metrics:

1. **average_time:** the average duration it takes to complete a request. It is computed by summing up the total time taken by all the concurrent requests and then dividing it by the number of concurrent requests.

2. **average_speed:** the rate at which the server processes tokens in the response. It is calculated by taking the average number of tokens received per request, and dividing it by average_time. The result is the average number of tokens processed per second.

For further details and to examine the scripts used, readers are encouraged to consult Appendix C.

STRUCTURED QUESTION TESTING

To further evaluate the 70b model's capabilities, we conducted a structured performance test involving three distinct categories of questions. The test comprised a total of 12 questions, segmented as follows:

1. **Standard Questions:** We posed five standard questions to the model. These questions were designed to assess the model's ability to handle typical queries.
2. **Complex Questions:** We challenged the model with two complex questions that required advanced reasoning or understanding. These questions were intended to test its capabilities in handling intricate and nuanced inquiries.
3. **Text-Based Questions:** Lastly, we presented the model with five questions that were directly related to a given text. This tested the model's comprehension skills and its ability to provide contextually relevant responses based on the provided material.

RESULTS

OUTPUT TOKEN TEST

The analysis of the performance data for the 8b and 70b models reveals several key insights into how these models handle varying workloads.

Figures 1 and 2 suggest that both models maintain reasonable performance levels when handling up to two concurrent requests, with the average time to complete a request increasing linearly as the output word count rises. However, when the number of concurrent requests is increased to four, there is a marked deterioration in performance, with a notably steeper increase in average completion time. This suggests that their current resource allocations are less effective at managing higher levels of request concurrency.

Figures 3 and 4 demonstrate that the average speed, measured in tokens per second, remains relatively stable across different output word counts for both the 8b and 70b LLaMa models. This indicates that the models are capable of maintaining consistent generation speeds regardless of the length of the output requested. However, a key observation from both figures is the decrease in average speed as the number of concurrent requests increases, which reflects a performance bottleneck when the models are tasked with handling multiple requests simultaneously.

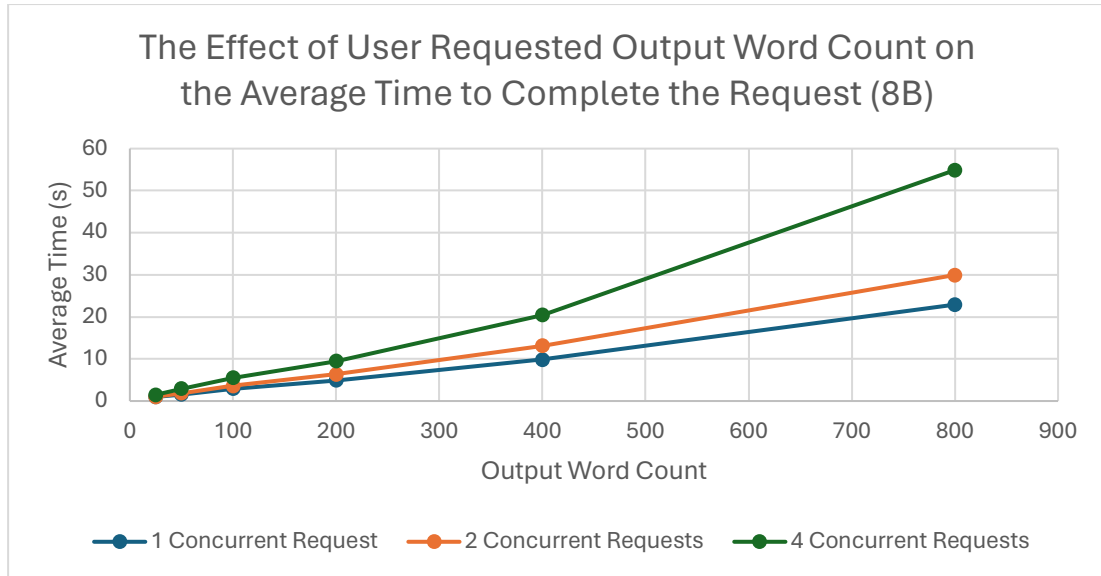


Figure 1: “Impact of Output Word Count on Average Time for Completion with Varying Concurrent Requests (LLaMa 8b Model). With 1 or 2 concurrent requests, the completion time increases by approximately 2.5 to 3.5 seconds for every additional 100 words. However, when handling 4 concurrent requests, the time increase is steeper, starting at 5 seconds per 100 words and then rising to about 8 seconds per 100 words once the output exceeds 400 words.”

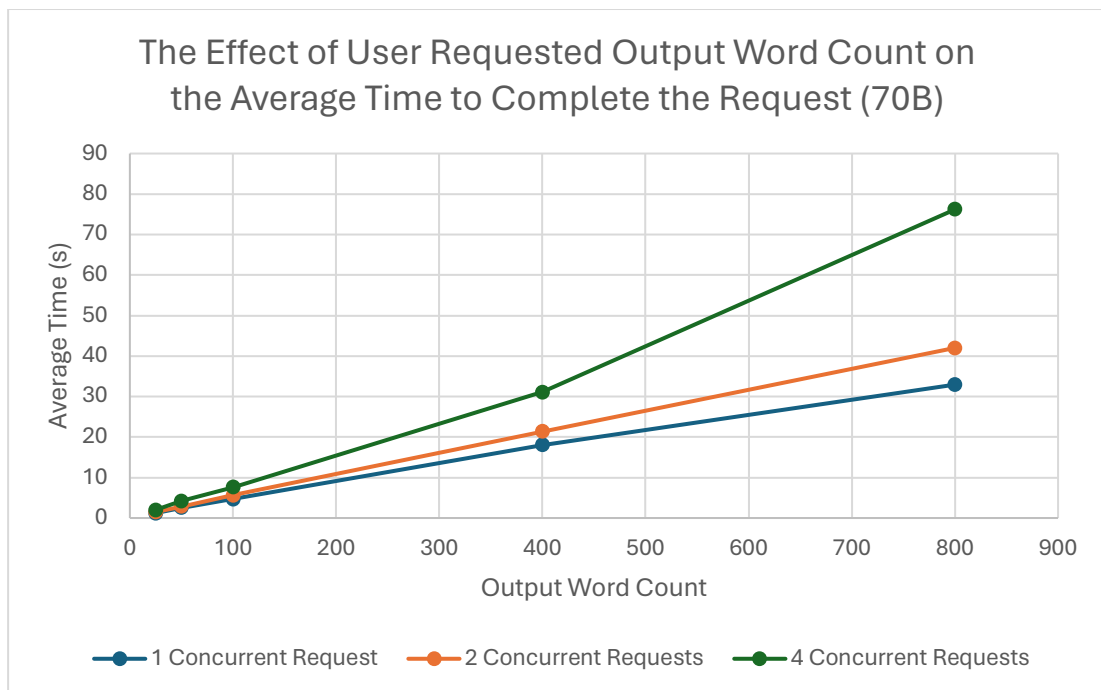


Figure 2: "Influence of Output Word Count on Average Request Completion Time Across Different Concurrent Request Scenarios (LLaMa 70b Model). With 1 or 2 concurrent requests, the completion time increases by approximately 4 to 5 seconds for every additional 100 words. However, when handling 4 concurrent requests, the time increase is steeper, starting at 7.5 seconds per 100 words and then rising to about 9.5 seconds per 100 words once the output exceeds 400 words.”

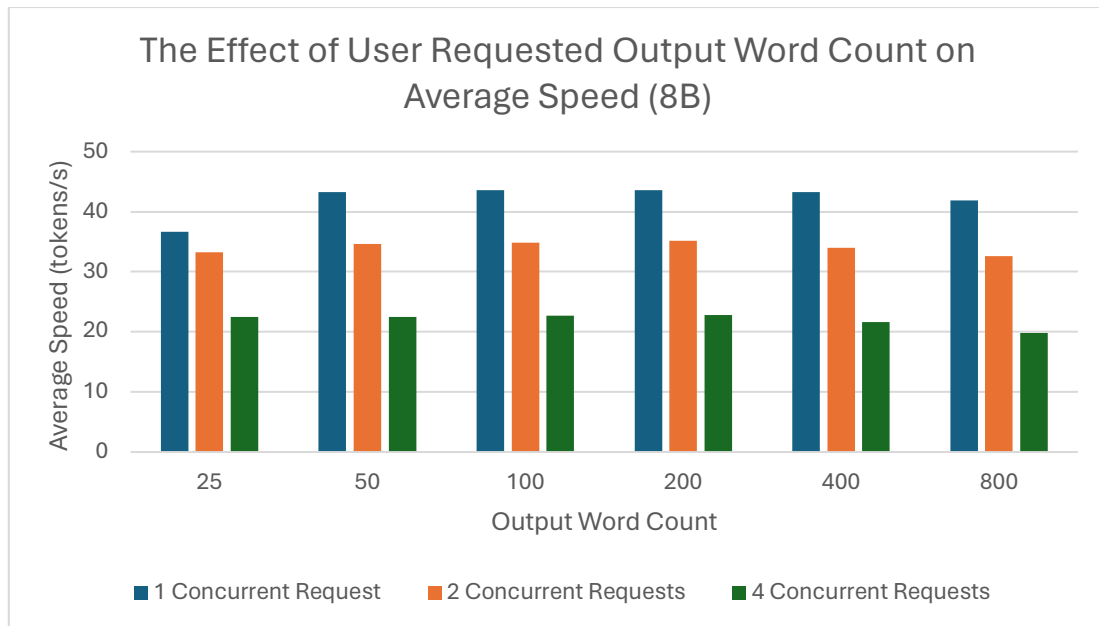


Figure 3. “Influence of Output Word Count on Average Speed Across Different Concurrent Request Scenarios (LLaMa 8b Model). When processing a single request, the average generation speed of the model is approximately 40 words per second. This speed declines to just over 30 words per second when managing 2 concurrent requests, and further drops to around 20 words per second when dealing with 4 concurrent requests.”

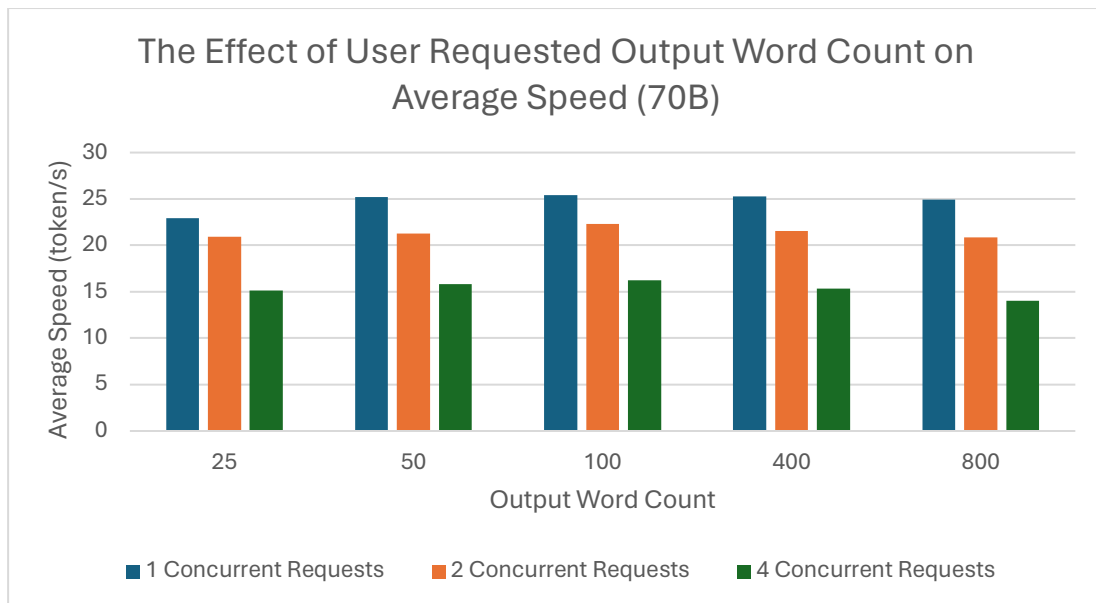


Figure 4. “Influence of Output Word Count on Average Speed Across Different Concurrent Request Scenarios (LLaMa 70b Model). When processing a single request, the average generation speed is approximately 25 words per second. This speed declines to just over 20 words per second when managing 2 concurrent requests, and further drops to around 15 words per second when dealing with 4 concurrent requests.”

Readers interested in examining the precise data points can refer to the tables in Appendix D. It should be noted that the output word count the model was instructed to adhere to may not exactly match the number of words generated by the model in every instance; however, the figures are sufficiently approximate for practical analysis.

INPUT TOKEN TEST

The table below contains the results from the input token test. It can be observed that the average time it takes to complete the request increases as the number of concurrent requests increases. However, this escalation in time is less pronounced compared to the results observed in the output token test. This observation aligns with the pricing structure of GPT-4, which supports the premise that output tokens carry a higher cost than input tokens, presumably due to the additional computational complexity and resources required for generating outputs.

Number of Requests	Output World Count	Average Tokens	Average Time (s)	Average Speed (tokens/s)
1	10	16	2.27796	7.02383
2	10	19	2.63539	7.20955
4	10	16.75	4.58825	3.65063

Table 4. “Input Token Test Results for LLaMa 3.1 70b”

SYSTEM AND GPU MEMORY UTILIZATION

System, CPU and GPU memory utilization were monitored throughout the performance testing using the commands detailed in Appendix E. The following conclusions were drawn based on our observed resource utilization metrics.

System Memory Utilization:

- If deploying a 40GB LLM, having more than 40GB of system memory is sufficient.
- The LLM is initially loaded into the system memory at a moderate pace of roughly 1GB per second. Subsequently, the LLM is rapidly transferred into the GPU memory.
- During concurrent request testing, system memory was virtually unused, suggesting that most of the heavy lifting is handled by the GPU.

System CPU Usage:

- Both single and concurrent access to the LLM resulted in only one CPU core being heavily utilized, operating at 80-100% capacity. The reasons for the single-core utilization and the specific computational tasks that the core is executing remain unclear.

GPU Memory Utilization:

- During concurrent testing, GPU memory usage remained constant at around 42GB, indicating that as long as the GPU memory exceeds the LLM size, it should be sufficient (see Figure 5).
- If two A100 80GB GPUs are available, Ollama will only utilize one GPU (see Figure 6). The implications and details of multi-GPU configurations were not explored in this study.

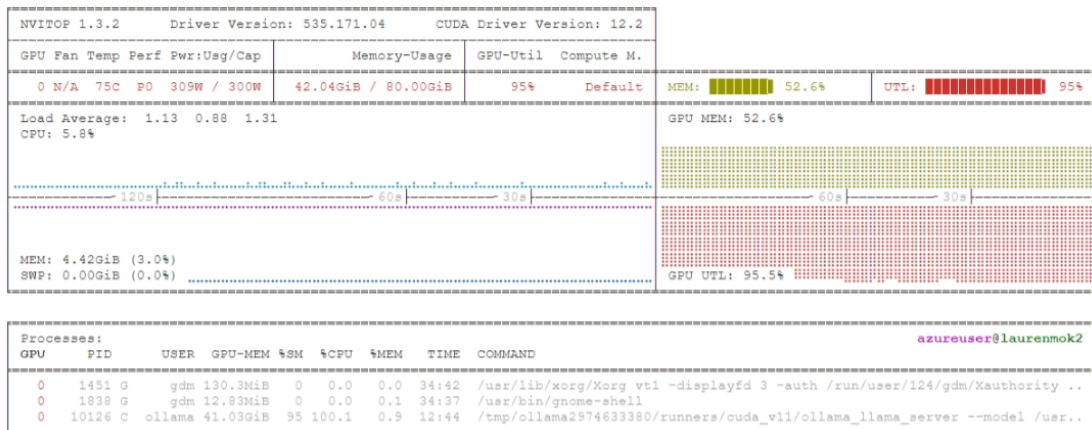


Figure 5. “GPU utilization metrics for 70b model on a single A100 GPU as captured by nvitop during performance testing”

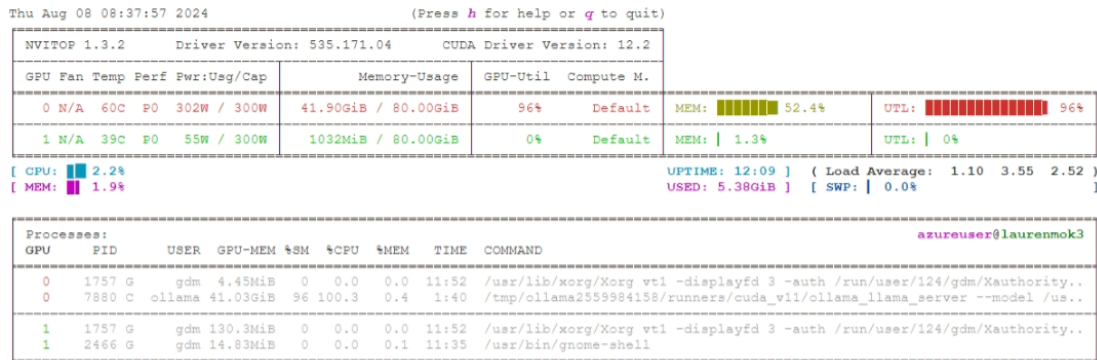


Figure 6. “GPU utilization metrics for 70b model on two A100 GPUs as captured by nvitop during output token performance testing. The metrics indicate that only one GPU was utilized.”

GPU Performance:

- When asking questions to the LLM, the GPU is almost fully loaded, confirming that the GPU determines the response speed.
- LLaMa can process a maximum of 4 concurrent requests; additional requests are queued, preventing out-of-memory issues (modifying the script to 8 concurrent requests shows that requests beyond the fourth are queued based on response times). Readers can refer to the results table from an initial performance testing trial in Appendix F to examine the individual response times for 8 concurrent requests.

Additional Observations:

- It is observed that although both input and output consume total tokens, the generation of output is slower.
- This can explain why with GPT-4, output tokens are more expensive than input tokens.
- The red boxes in Figure 7 highlight key data:
 - After running "ollama pull," there is a significant amount of buffer/cache (44730.3 MiB).

- During the "ollama run," the GPU memory shows 685MiB / 81920MiB used before running, and after the run, it shows 42181MiB / 81920MiB used, indicating that the LLM has been loaded into the GPU memory.

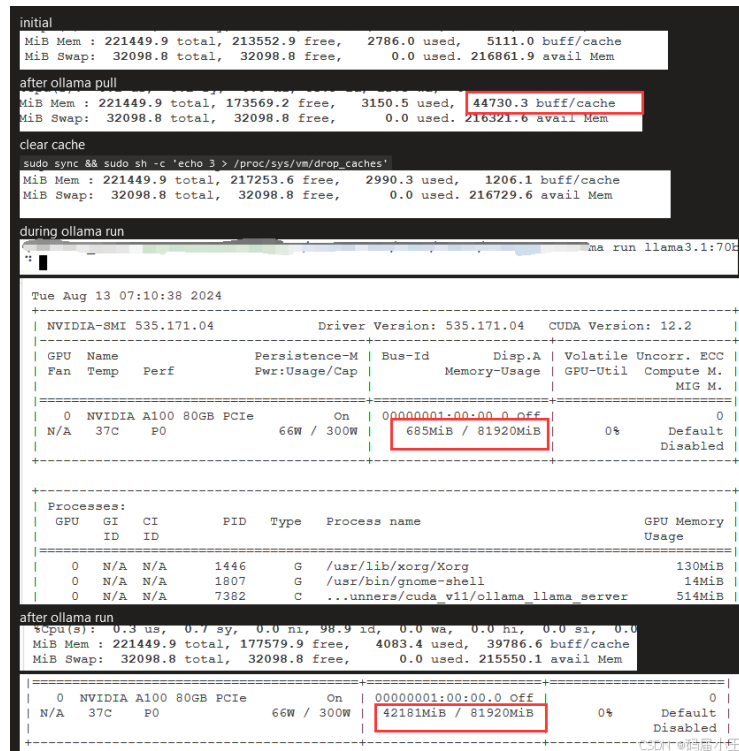


Figure 7. "Resource utilization metrics while pulling and running Ollama"

STRUCTURED QUESTION TESTING

The 70b model delivered highly accurate and coherent responses across all categories of questions. In the text-based question category, the model was subjected to a scenario with 2 concurrent requests to assess its efficiency under simultaneous processing conditions. It was observed that the response time for the initial request was marginally quicker, with the subsequent request experiencing a slight delay, typically by a few seconds. The questions posed to the model, along with the corresponding output answers generated by the model, can be found in Appendix G.

DISCUSSION

The analysis of the performance data and system resource utilization has yielded valuable insights into the functioning and requirements of the LLaMa3.1-8b and LLaMa3.1-70b models. Our findings indicate that while both models perform well under low to moderate request concurrency, there is a notable degradation in performance as the number of concurrent requests increases. This is evidenced by the longer average times and reduced average speeds observed during testing.

With regards to multi-GPU usage, it is interesting to note that even with two A100 GPUs available, Ollama only utilized one. This finding suggests that there is potential for further optimization in the use of multiple GPUs, which could potentially lead to better performance and resource utilization. Further research could explore how to effectively distribute the workload across multiple GPUs to enhance the models' efficiency, particularly under high concurrency conditions.

The observation that only a single CPU core is heavily utilized, regardless of the number of concurrent requests, raises questions about the potential bottlenecks in the CPU's involvement in processing requests. Understanding why the models do not leverage multiple cores and how this affects performance could be a significant area for future research.

Based on our findings, we can make several recommendations for practitioners looking to scale their use of the LLaMa models. For instance, a single A100 GPU appears to be sufficient for handling up to two concurrent requests efficiently. Organizations could start with this configuration and scale their resources as necessary, depending on the demand and performance requirements. Looking forward, there is a need to establish guidelines for the allocation of GPU resources for handling a greater number of concurrent requests. This would help organizations plan their infrastructure investments more effectively and avoid over-provisioning, thereby reducing costs while maintaining optimal performance.

CONCLUSION

In conclusion, our investigation into the performance and resource demands of the LLaMa3.1-8b and LLaMa3.1-70b models has highlighted the importance of understanding and optimizing the computational resources required for these advanced AI tools. By conducting thorough performance testing and monitoring system resource usage, we have established practical guidelines for the efficient and cost-effective deployment of these models. Our findings indicate that a single A100 GPU is capable of handling up to two concurrent requests effectively, and can process a maximum of four concurrent requests. To avoid out-of-memory errors, any requests beyond this limit are placed into a queue for sequential processing. Our research serves as a valuable resource for researchers and practitioners in the field of artificial intelligence, enabling them to make informed decisions on resource allocation and scaling to leverage the full potential of large language models like LLaMa.

Appendix A: Ollama and GPU Specifications

NVIDIA T4 Specifications

Performance	<div>Turing Tensor Cores</div> <div>320</div> <div>NVIDIA CUDA® cores</div> <div>2,560</div> <div>Single Precision Performance (FP32)</div> <div>8.1 TFLOPS</div> <div>Mixed Precision (FP16/FP32)</div> <div>65 FP16 TFLOPS</div> <div>INT8 Precision</div> <div>130 INT8 TOPS</div> <div>INT4 Precision</div> <div>260 INT4 TOPS</div>
Interconnect	<div>Gen3</div> <div>x16 PCIe</div>
Memory	<div>Capacity</div> <div>16 GB GDDR6</div> <div>Bandwidth</div> <div>320+ GB/s</div>
Power	<div>70 watts</div>

NVIDIA A100 Specifications

	A100 80GB PCIe	A100 80GB SXM
FP64	9.7 TFLOPS	
FP64 Tensor Core	19.5 TFLOPS	
FP32	19.5 TFLOPS	
Tensor Float 32 (TF32)	156 TFLOPS 312 TFLOPS*	
BFLOAT16 Tensor Core	312 TFLOPS 624 TFLOPS*	
FP16 Tensor Core	312 TFLOPS 624 TFLOPS*	
INT8 Tensor Core	624 TOPS 1248 TOPS*	
GPU Memory	80GB HBM2e	80GB HBM2e
GPU Memory Bandwidth	1,935 GB/s	2,039 GB/s
Max Thermal Design Power (TDP)	300W	400W ***
Multi-Instance GPU	Up to 7 MIGs @ 10GB	Up to 7 MIGs @ 10GB
Form Factor	PCIe Dual-slot air-cooled or single-slot liquid-cooled	SXM
Interconnect	NVIDIA® NVLink® Bridge for 2 GPUs: 600 GB/s ** PCIe Gen4: 64 GB/s	NVLink: 600 GB/s PCIe Gen4: 64 GB/s
Server Options	Partner and NVIDIA-Certified Systems™ with 1-8 GPUs	NVIDIA HGX™ A100-Partner and NVIDIA-Certified Systems with 4,8, or 16 GPUs NVIDIA DGX™ A100 with 8 GPUs

Appendix B: Running LLaMa 3.1 Locally with Ollama

Ollama Website: [Ollama](#)

Ollama GitHub: [GitHub - ollama/ollama: Get up and running with Llama 3.1, Mistral, Gemma 2, and other large language models.](#)

Instructions for Running a LLaMa model using Ollama: [How to Run Open Source LLMs Locally Using Ollama \(freecodecamp.org\)](#)

Instructions for Creating a Compute Instance on Azure:

Open your workspace on Microsoft Azure Machine Learning Studio. Create a compute instance as illustrated in figures 1a and 1b, with the virtual machine type set to ‘GPU’. If testing LLaMa 3.1 70B, set the size to an A100 GPU, such as the ‘Standard_NC24ads_A100_v4’.

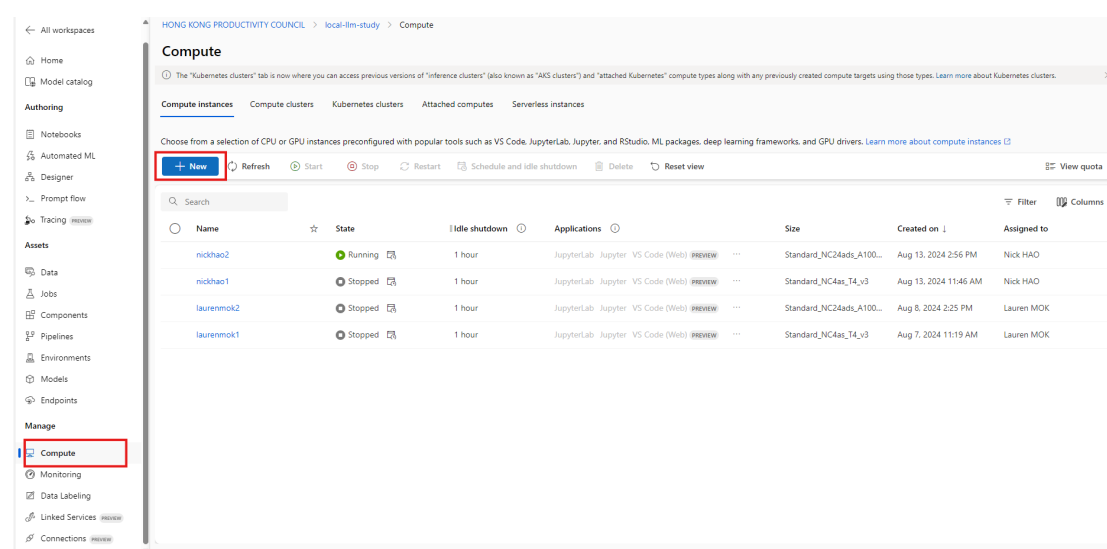


Figure 8. “Creating a new compute instance on Azure”

Create compute instance

Warning: We are unable to verify your virtual network settings due to insufficient permissions. If the workspace storage account is behind a virtual network, make sure to select the same virtual network for this compute instance to prevent connectivity issues.

1 Required settings

2 Scheduling optional

3 Security optional

4 Applications optional

5 Environments optional

6 Tags optional

7 Review

Configure required settings
Select the name and virtual machine size you would like to use for your compute instance

Note that a compute instance can not be shared. It can only be used by a single assigned user. By default, it will be assigned to the creator and you can change this to a different user in the Security step.

Compute name *

Virtual machine type ☐ CPU ☒ GPU

Virtual machine size ☒ Select from recommended options ☐ Select from all options

Name ↑	GPU device	Workload types	Available quota ⓘ	Cost ⓘ
<input checked="" type="checkbox"/> Standard_NC24ads_A100_v4 24 cores, 220GB RAM, 64GB storage		Single-node, high-scale training of deep learning models	48 cores	\$3.67/hr

Review + Create Back Next Cancel

Figure 9. “Selecting your virtual machine type and size for your compute”

Resource properties

Status
Running

Last operation
Created at Aug 7, 2024 11:20 AM: Succeeded

Virtual machine size
Standard_NC4as_T4_v3 (4 cores, 28 GB RAM, 176 GB disk)

Processing unit
GPU - 1 x NVIDIA Tesla T4

Estimated cost
\$0.53/hr (when running)

Additional data storage
--

Applications
JupyterLab Jupyter VS Code (Web) PREVIEW VS Code (Desktop) PREVIEW **Terminal** Notebook

Created on
7/8/2024, 11:19:58 am

SSH access
Disabled

Root access
Enabled

Figure 10. “Opening a Terminal window”

Once the compute has been created, open a Terminal window (Figure 1c) and run the following commands to install ollama and pull the llama 3.1-70B model:

```
curl -fsSL https://ollama.com/install.sh | sh
```

```
ollama pull llama3.1:70b
```

```
ollama run llama3.1:70b
```

Appendix C: Python Scripts for Testing Model Performance

Input Token Script:

[ai_script/A100_80G_Llama3_70B/70b_input-stream.py at main · codeHui/ai_script · GitHub](#)

Output Token Script:

[ai_script/A100_80G_Llama3_70B/70b_output-stream.py at main · codeHui/ai_script · GitHub](#)

Appendix D: Output Token Test Results

Number of Requests	Output World Count	Average Tokens	Average Time (s)	Average Speed (tokens/s)
1	25	36	0.983328	36.6104
1	50	68	1.57218	43.2521
1	100	127	2.91746	43.531
1	200	215	4.92939	43.6159
1	400	426	9.85016	43.248
1	800	960	22.9157	41.8926
2	25	33.5	1.0074	33.2538
2	50	64	1.84838	34.6249
2	100	126	3.61762	34.8295
2	200	224	6.37752	35.1233
2	400	446.5	13.1239	34.0218
2	800	978	29.9562	32.6477
4	25	33.5	1.48696	22.5291
4	50	66.5	2.95279	22.5211
4	100	125.5	5.53904	22.6573

Number of Requests	Output World Count	Average Tokens	Average Time (s)	Average Speed (tokens/s)
4	200	215.25	9.44115	22.7991
4	400	441.75	20.4161	21.6374
4	800	1090.5	54.8982	19.864

Table 2. “Output Token Test Results for LLaMa 3.1 8b”

Number of Requests	Output World Count	Average Tokens	Average Time (s)	Average Speed (tokens/s)
1	25	28	1.22156	22.9214
1	50	63	2.49925	25.2076
1	100	119	4.68078	25.4231
1	400	456	18.0439	25.2717
1	800	821	32.9355	24.9275
2	25	31	1.4789	20.9615
2	50	60.5	2.84061	21.2983
2	100	125	5.59961	22.323
2	400	460.5	21.3357	21.5835
2	800	877	42.0024	20.8798
4	25	30.5	2.0107	15.1689
4	50	66.5	4.21136	15.7906
4	100	122.75	7.56207	16.232
4	400	478.25	31.1035	15.3761
4	800	1066.75	76.1915	14.0009

Table 3. “Output Token Test Results for LLaMa 3.1 7b”

Appendix E: Tools for Monitoring Resource Usage

CPU and memory monitoring tools

1) **top**

- a) Displays the resource usage of each process in the system in real time, including CPU, memory, etc.
- b) Run the command: `top`

2) **htop**

- a) An enhanced version of `top`, providing a more friendly interface and more functions.
- b) Run the command: `htop`

3) **vmstat**

- a) Provides statistics about system processes, memory, paging, block IO, traps, and CPU activity.
- b) Run the command: `vmstat 1` (updates once per second)

4) **free**

- a) Displays the system memory usage.
- b) Run the command: `free -h` (displayed in human readable format)

Nvidia GPU Monitoring Tools

1) **nvidia-smi**

- a) Displays GPU usage, video memory usage, temperature and other information.
- b) Run the command: `nvidia-smi`

2) **watch nvidia-smi**

- a) Refreshes the `nvidia-smi` output once per second
- b) Run the command: `watch -n 1 nvidia-smi`

Comprehensive monitoring tools

1) **glances**

- a) `glances` It is a cross-platform system monitoring tool that can display CPU, memory, disk, network and other system information.
- b) Installation command: `sudo apt-get install glances`(Debian/Ubuntu), `sudo yum install glances`(CentOS/RHEL)
- c) Run the command: `glances`

2) **nvitop**

- a) `nvitop` is a Python tool specifically designed to monitor Nvidia GPUs, similar to Nvidia GPUs `htop`.
- b) Run the command: `nvitop`

Appendix F: Performance Testing - Individual Response Times

70b Model – 1x A100 GPU					
Concurrent Requests	Word Count	Average Response Time (seconds)	Sum Response Time (seconds)	Response Times (seconds)	Load Duration (seconds)
1	50	3.0019	3.0019	[2.996571698]	[0.019734111]
1	100	5.9131	5.9131	[5.908149849]	[0.02093999]
1	400	19.5633	19.5633	[19.557673881]	[0.019956508]
1	3000	66.9737	66.9737	[66.968428784]	[0.018935217]
2	100	5.8449	11.6898	[5.626313133, 6.053264504]	[0.020244471, 0.062354101]
2	400	21.9475	43.8950	[20.331652885, 23.553933646]	[0.018858932, 0.018883152]
2	800	38.6704	77.3408	[38.296151575, 39.034841958]	[0.018437744, 0.018369358]
4	100	6.4846	25.9385	[5.625726439, 5.835892537, 6.953059487, 7.505292355]	[0.13941773, 0.223227117, 0.020247624, 0.063411136]
4	400	29.3982	117.5929	[27.118555976, 27.293136539, 30.739015291, 32.422647333]	-
8	800	89.9706	719.7651	[43.109153096, 54.28461106, 60.968316952, 79.976316833, 89.292250087, 117.27595815, 137.199554638, 137.622542002]	[0.021297854, 0.105603716, 0.149433275, 0.062230551, 0.15812038, 0.309334145, 0.421544461, 0.48951834]

8b Model – 1x T4 GPU					
Concurrent Requests	Word Count	Average Response Time (seconds)	Sum Response Time (seconds)	Response Times (seconds)	Load Duration (seconds)
4	50	2.8192	11.2767	[2.68595384, 2.727811876, 2.896866042, 2.945711403]	[0.092141998, 0.133172995, 0.213177761, 0.257167853]
10	100	8.6930	-	[3.42685148, 4.35198224, 5.832711984, 6.180287143, 8.793814522, 9.436861302, 10.361947618, 11.890747776]	[0.033431092, 0.191193256, 0.0865689, 0.168664533, 0.42973947, 0.233835447, 0.32603342, 0.377648379]

				13.179860041, 13.281433189]	0.468099862, 0.520667551]
5	100	5.8111	-	[4.373233084, 5.45082269, 5.683002776, 5.920354073, 7.553693428]	[0.077269015, 0.115848774, 0.030078306, 0.156123567, 0.152486319]
2	100	3.8486	-	[3.806278125, 3.8841858]	[0.021744261, 0.062289575]
2	800	26.6537	53	[25.180122152, 28.120668329]	[0.063975744, 0.021647797]
4	400	24.2521	97.0084	[22.227794301, 22.891042265, 25.517087434, 26.328928608]	[0.022598784, 0.027032603, 0.021490781, 0.021347384]
4	800	73.2556	293.0225	[60.774537858, 65.009188426, 68.872913162, 98.315949133]	[0.024492331, 0.031434118, 0.02285158, 0.026812139]
8	800	86.3788	691.0308	[44.884686175, 49.105129155, 50.593365034, 56.539519702, 113.066071365, 120.741595305, 122.325417532, 133.678898617]	[0.211956364, 0.042866107, 0.074064392, 0.14399556, 0.342576944, 0.439575642, 0.227407555, 0.485938489]
1	3000	15.4697	15.4697	[15.466842]	[2.647406898]
4	3000	59.2143	236.8574	[41.995455074, 42.975652793, 51.782464306, 100.083169975]	[0.063777615, 0.263891024, 0.022250485, 0.151812941]
7	10000	75.8739	531.1172	[45.330521687, 45.866410174, 50.71598573, 58.108327401, 104.803186544, 105.642315508, 120.604181087]	[0.338648019, 0.215993005, 0.026795145, 0.06794266, 0.41800832, 0.496010952, 0.353171325]

Appendix G: Questions Used to Test Model Performance

Normal Questions:

1. What is the capital city of France?

Output: The capital city of France is Paris.

2. Who wrote the play "Romeo and Juliet"?

Output: William Shakespeare wrote the play "Romeo and Juliet".

3. What is the chemical symbol for water?

Output: H₂O is the chemical symbol for water.

4. In what year did the first manned moon landing occur?

Output: The first manned moon landing occurred in 1969.

5. What is the process by which plants make their own food using sunlight called?

Output: Photosynthesis is the process by which plants make their own food using sunlight.

Demanding Questions:

1. Explain the concept of quantum entanglement and its significance in the field of quantum mechanics.

Output:

A question that gets to the heart of some pretty mind-bending physics!

Quantum entanglement is a phenomenon in which two or more particles become correlated in such a way that the state of one particle cannot be described independently of the others, even when they are separated by large distances.

In other words, if something happens to one particle, it instantly affects the state of the other entangled particles, regardless of how far apart they may be. This effect happens even faster than the speed of light, which is why Einstein called it "spooky action at a distance."

Entanglement occurs when two or more particles interact in such a way that their properties, like spin or momentum, become linked together.

When this happens, measuring the state of one particle instantly determines the state of the other entangled particles.

The significance of quantum entanglement lies in its implications for our understanding of reality and the behavior of matter at the smallest scales:

1. **Non-locality:** Entanglement shows that the information about a particle's state can be instantaneously transmitted to another particle, regardless of distance. This challenges classical notions of space and time.
2. **Quantum computing:** Entangled particles are used as qubits (quantum bits) in quantum computers, enabling them to perform calculations exponentially faster than classical computers for certain problems.
3. **Secure communication:** The no-cloning theorem, which is a consequence of entanglement, ensures that any attempt to eavesdrop on an encrypted message will be detectable, making entanglement-based quantum cryptography theoretically unbreakable.
4. **Fundamental limits of measurement:** Entanglement highlights the strange implications of wave function collapse and the role of observation in shaping reality.

In summary, quantum entanglement is a fascinating phenomenon that demonstrates the non-intuitive nature of quantum mechanics and has significant implications for our understanding of space, time, and matter at the smallest scales.

-
2. How would you approach solving the problem of climate change on a global scale, considering both technological and socio-political factors?

Output:

Addressing climate change requires a comprehensive approach that incorporates technological innovation, policy changes, and societal transformations. Here's a step-by-step plan to tackle this complex issue:

Short-term goals (2025-2035)

1. **Transition to renewable energy:** Invest in solar, wind, hydro, and geothermal power to reduce dependence on fossil fuels. Implement policies like carbon pricing, tax incentives, and net-metering laws to encourage adoption.
2. **Energy efficiency:** Develop and deploy energy-efficient technologies for buildings, transportation, and industry. Implement standards for appliance efficiency, building insulation, and smart grids.
3. **Electrify transportation:** Promote electric vehicles through subsidies, infrastructure development (e.g., charging stations), and regulatory support.
4. **Carbon capture and storage:** Develop and deploy carbon capture technology for industrial sources like cement factories and power plants.

5. **Sustainable land use:** Implement policies to prevent deforestation, promote reforestation, and adopt sustainable agriculture practices.

Mid-term goals (2035-2050)

1. **Global cooperation:** Establish a binding international agreement on climate change mitigation and adaptation efforts.
2. **Carbon pricing:** Implement a global carbon pricing mechanism to provide a financial incentive for reducing emissions.
3. **Research and development:** Invest in R&D for new clean energy technologies, advanced nuclear power, and carbon utilization strategies.
4. **Climate-resilient infrastructure:** Develop and deploy climate-resilient infrastructure, such as sea walls, levees, and green roofs.
5. **Education and awareness:** Launch global education campaigns to raise awareness about climate change causes, effects, and solutions.

Long-term goals (2050-2100)

1. **Net-zero emissions:** Achieve net-zero greenhouse gas emissions globally through a combination of technological advancements, behavioral changes, and policy measures.
2. **Carbon removal technologies:** Develop and deploy carbon dioxide removal technologies like direct air capture, afforestation/reforestation, and soil carbon sequestration.
3. **Sustainable consumption patterns:** Promote sustainable lifestyles, reduce waste, and encourage sharing economies.
4. **Climate-resilient cities:** Design and develop climate-resilient cities with green infrastructure, flood protection measures, and adaptive governance structures.
5. **Global climate governance:** Establish a robust global climate governance framework to ensure coordinated action, accountability, and transparency.

Socio-political factors

1. **International cooperation:** Foster global cooperation through diplomatic efforts, economic incentives, and knowledge sharing.
2. **Climate justice:** Address the disproportionate impact of climate change on vulnerable populations through equitable policies and adaptation measures.
3. **Public engagement:** Engage citizens in climate decision-making processes to ensure their voices are heard and concerns addressed.
4. **Economic diversification:** Encourage economic diversification to reduce dependence on fossil fuels and promote low-carbon industries.
5. **Climate education and training:** Develop climate-focused educational programs for various sectors, including industry, agriculture, and urban planning.

Technological factors

1. **Clean energy innovation:** Accelerate the development and deployment of clean energy technologies like advanced solar panels, fuel cells, and hydrogen production.
2. **Energy storage:** Improve energy storage solutions to enable greater integration of intermittent renewable sources into the grid.
3. **Smart grids and infrastructure:** Develop and deploy smart grid technologies to optimize energy distribution and consumption.
4. **Carbon capture and utilization:** Advance carbon capture and utilization technologies to convert CO₂ into valuable products like fuels, chemicals, or building materials.
5. **Artificial intelligence and data analytics:** Leverage AI and data analytics to optimize climate mitigation strategies, predict extreme weather events, and identify areas of high vulnerability.

Implementation

1. **Global coordination:** Establish a global coordinating body to facilitate cooperation, knowledge sharing, and progress tracking.
2. **National action plans:** Encourage countries to develop and implement national action plans for climate change mitigation and adaptation.
3. **Sectoral initiatives:** Launch sector-specific initiatives (e.g., energy, transportation, agriculture) to drive innovation and adoption of low-carbon technologies.
4. **Private sector engagement:** Engage the private sector in climate action through incentives, partnerships, and regulatory frameworks.
5. **Monitoring and evaluation:** Establish a robust monitoring and evaluation framework to track progress, identify areas for improvement, and adjust strategies as needed.

Addressing climate change requires a sustained effort from governments, industries, civil society, and individuals worldwide. By combining technological innovation with socio-political transformation, we can create a more sustainable future for all.

Questions based on Provided Information: (provided information and model outputs have been hidden for data privacy)

1. Based on the below information, how many annual leaves are employees allowed?

concurrent_requests: 2

Average response time: 6.5895 seconds

sum response time: 13.1789 seconds

total_duration: [6.261619539, 6.906818389]

load_duration: [0.059278025, 0.018438285]

2. Based on the below information, if my project estimate is HK \$200,000, what authority do I need approval from? If the project is cross-divisional, who needs to sign the approval?

concurrent_requests: 2
Average response time: 6.4952 seconds
sum response time: 12.9904 seconds
total_duration: [4.863672925, 8.117569459]
load_duration: [0.062695877, 0.01974573]

3. Based on the below information, what steps do I need to take if I want to request access to personal data?

concurrent_requests: 2
Average response time: 7.3608 seconds
sum response time: 14.7215 seconds
total_duration: [6.095342452, 8.616791385]
load_duration: [0.061669842, 0.019890898]

4. Based on the below information, are employees at HKPC allowed to download external applications on their work computers?
media (which could include downloaded applications) are only allowed if they have been checked for viruses first.

concurrent_requests: 2
Average response time: 8.8708 seconds
sum response time: 17.7416 seconds
total_duration: [8.299322537, 9.432749814]
load_duration: [5.195784428, 5.195050756]

5. Based on the below information, what should an employee do if they accidentally clicked on a link provided by a phishing email and entered their username and password?

concurrent_requests: 2
Average response time: 10.3628 seconds
sum response time: 20.7257 seconds
total_duration: [9.903586812, 10.812611548]
load_duration: [4.995614506, 4.955380326]

6. 如果要 book classroom, 找邊個?

concurrent_requests: 2

Average response time: 13.4794 seconds

sum response time: 26.9587 seconds

total_duration: [12.548170142, 14.401124952]

load_duration: [4.995667662, 4.952055798]