Resource Management for GPT-based Model Deployed on Clouds: Challenges, Solutions, and Future Directions

Yongkang Dang[1], Yiyuan He[1,2], Minxian Xu[1,✉], and Kejiang Ye[1]

[1] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, China
[2] Southern University of Science and Technology, Shenzhen, Guangdong, China
{yk.dang, yy.he2, mx.xu, kj.ye}@siat.ac.cn

Abstract. The widespread adoption of the large language model (LLM), e.g. Generative Pre-trained Transformer (GPT), deployed on cloud computing environment (e.g. Azure) has led to a huge increased demand for resources. This surge in demand poses significant challenges to resource management in clouds. This paper aims to highlight these challenges by first identifying the unique characteristics of resource management for the GPT-based model. Building upon this understanding, we analyze the specific challenges faced by resource management in the context of GPT-based model deployed on clouds, and proposed resource profiling and prediction algorithms for inference requests. To facilitate efficient resource management, we introduce a comprehensive resource management framework, featuring resource profiling and forecasting methods specifically designed for GPT-based models. Furthermore, we discuss the future directions for resource management for the GPT-based model, highlighting potential areas for further exploration and improvement. Through this study, we aim to provide valuable insights into resource management for GPT-based models deployed in clouds and promote their sustainable development for GPT-based models and applications.

Keywords: GPT-based Model, Cloud Computing, Task Profiling, Resource Management Framework

1 Introduction

The GPT-based model is a language generation model based on the transformer architecture [16], which learns the statistical regularities and semantic knowledge of language through unsupervised pre-training on large-scale text datasets. The model can then be fine-tuned on specific domain or task data through supervised or semi-supervised learning to adapt to different language application scenarios [3, 13]. The GPT-based model can generate natural, fluent, context-aware, and semantically coherent language content, making it suitable for applications such as text summarization, machine translation, sentiment analysis, question answering systems, and chatbots. Figure 1 provides examples of current applications

of the GPT-based model for different areas [2, 11], including academic, medical, office, education and marketing.

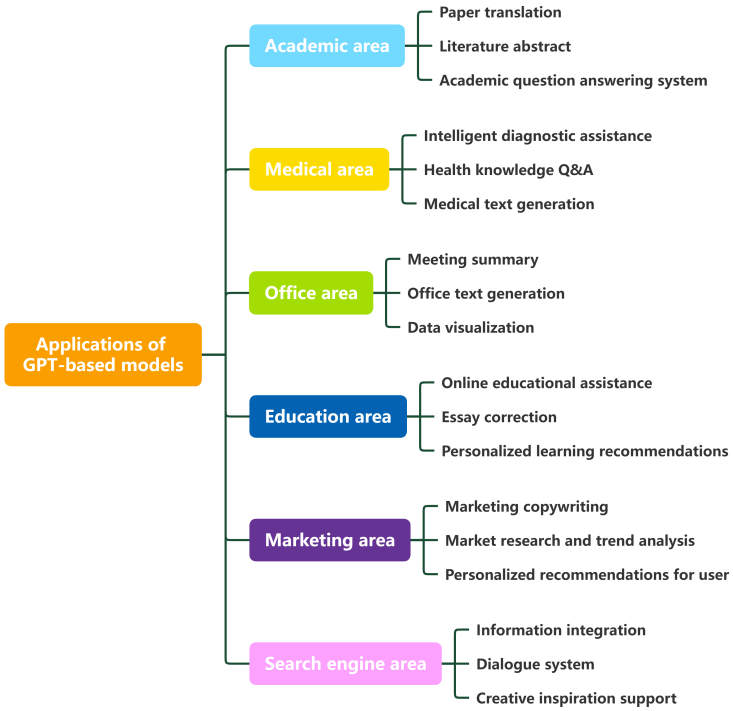


Fig. 1: GPT-based Model Application Areas

The GPT-based model was first proposed and developed by OpenAI and has been developed into multiple versions and variants, such as GPT-1, GPT-2, GPT-3, and GPT-4 [12]. The main differences between these models lie in the number of parameters, dataset size, and training methods. For example, GPT-3 is currently one of the largest GPT-based models, with 175 billion parameters, requiring 800 GB of storage [13]. It has achieved excellent performance on multiple natural language processing tasks. Table 1 lists several mainstream GPT-based models, providing relevant details such as model names, research teams, release dates, and model sizes, with each row representing a distinct model.

The rapid development and widespread application of GPT-based model have led to the increased demand for resources, and the current GPT-based model has been deployed on public clouds like Azure and Google Cloud for training and inference, as GPT-based models are typically too large and resource-intensive to be deployed on edge devices or small-scale hardware [8]. Therefore, they are better suited for cloud-based deployments, which also makes the resource management for GPT-based model facing some specific challenges. In order to summarize these challenges and propose corresponding solutions, in this section, we

Table 1: Parameters of Mainstream GPT-based Models [23]

Model	Research Teams	Release Date	Size
GPT-4	OpenAI	14-Mar-23	1760B
GPT-3	OpenAI	28-May-20	175B
PanGu-α	Pengcheng Laboratory	26-Apr-21	207B
GLM	ZhiPuAI	17-Mar-22	130B
Yi-1.5	01-ai	May-24	34B
Llama3	Meta Llama	18-April-24	70B
Claude2	Anthropic	11-July-22	52B
OPT-175B	Meta	3-May-22	175B
PaLM	Google	Apr-22	540B
ERNIE Bot	Baidu	14-Mar-23	17B
MT-NLG	Microsoft and NVIDIA	11-Oct-21	530B
Gopher	DeepMind	8-Dec-21	280B
CPM-2	Beijing Academy of Artificial Intelligence	Jun-21	198B
GPT-Neo-X-20B	EleutherAI	Apr-22	20B

will identify the unique characteristics of resource management for GPT-based models and establish evaluation metrics for this specific domain.

1.1 Unique Characteristics of Resource Management for GPT-based Model

Through extensive research and compared with traditional applications, we identify the following unique characteristics of resource management for GPT-based model:

Large-scale computational demands due to huge amount of parameters and fine-tuning: The GPT-based model typically consists of billions of parameters, necessitating a substantial amount of computational resources during both training and inference processes [4]. Training GPT-based models typically requires specialized hardware such as Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs) due to the sheer number of calculations involved. In addition, fine-tuning a pre-trained GPT-based model on a specific task requires additional compute resources, as the model needs to adapt to the task through further training [7]. This complexity makes resource management more intricate, requiring efficient allocation and utilization of computational resources to ensure the model operates efficiently.

High storage demands to support rapid data access: The GPT-based model's large parameter size requires significant storage space to accommodate model parameters and intermediate computation results. Running these models can quickly consume all available memory on conventional hardware [17]. Therefore, resource management must consider how to effectively manage storage resources to meet the model's requirements while ensuring rapid data access and processing.

High-speed network demands to enable efficient parallelism: During model training, the GPT-based model handles vast datasets and performs complex computations and parameter optimization, demanding fast data transmission and stable network connectivity [10]. In the inference phase, efficient

network resource utilization directly impacts inference speed and response time. The GPT-based model needs to generate outputs based on inputs and provide results in real-time or near real-time conditions. Hence, network resources are crucial for achieving fast responses and efficient inference.

Long training and inference processes than traditional AI models: Traditional AI applications often have lower computational requirements and faster inference time. However, due to the complexity and scale of the GPT-based model, its training and inference processes typically require extended periods. Resource management must consider how to maintain system stability and performance over an extended duration while ensuring the rational allocation and utilization of resources.

Dynamic resource demands from varied complexity of tasks: In practical applications, the resource requirements of the GPT-based model may vary over time and across different tasks [6], e.g. machine translation, text summarization and question answering. These tasks can have dynamic resources demand due to the different degree of complexity (e.g. difficulty of questions and expected output length). Resource management must possess dynamic adjustment capabilities, allowing for the dynamic allocation of computational and storage resources based on actual demands to adapt to different stages and tasks.

By understanding and addressing these unique characteristics, effective resource management strategies can be developed to ensure optimal performance and utilization of the GPT-based model in various applications.

1.2 Evaluation Metrics for Resource Management for GPT-based Model

In order to effectively evaluate the resource management of GPT-based model, we can consider the following metrics:

Resource Utilization: It refers to the degree to which the model effectively utilizes available resources during the training or inference process. For the GPT-based model, resources primarily include computational resources (such as CPUs and GPUs), storage resources (such as memory and disk space), and network resources. Evaluating resource utilization involves ensuring that the model maximizes the use of available resources to improve efficiency and minimize resource waste. This can be achieved through optimization of scheduling algorithms and parallel computing techniques. Higher resource utilization indicates efficient utilization of computational, storage, and network resources, enhancing overall system performance.

Time Efficiency: It refers to the time taken by the model to complete a set of given tasks (e.g. makespan metric used in traditional task scheduling). For the GPT-based models, time efficiency includes both model training time and inference time. During training, time efficiency focuses on the speed of parameter updates on a given dataset[18]. Training time efficiency can be improved through optimization of scheduling algorithms, distributed training, and hardware acceleration. During inference, time efficiency concerns the speed at which the model processes input data and generates outputs. Inference time directly

affects the real-time performance and responsiveness of the model in practical applications. Techniques such as parallel computing, batch processing, and hardware optimization can improve inference time efficiency. Higher time efficiency means the model can complete training and inference tasks more quickly, thereby increasing overall production efficiency.

Cost Efficiency: The cost of the GPT-based model mainly includes computational cost, storage cost, and network transmission cost [20, 22]. Computational cost evaluation primarily considers the model's complexity, workload, and hardware used. Lower computational costs reduce resource investments for specific tasks. Storage cost involves expenses for storing model parameters, intermediate results, and cached data, measured by the model's size and storage capacity. Lower storage costs reduce resource demands. Network transmission cost covers expenses for data transmission during training or inference, including model parameters and results. Lower network costs indicate efficient use of network resources, reducing transmission time and bandwidth expenses.

This paper aims to highlight the specific challenges in resource management for GPT-based models and propose corresponding solutions. Our work positions as a review paper, compared with existing review, survey and taxonomy papers related to surveying LLM model mechanisms [19, 21, 23], we focus on resource management perspective for LLMs. To the best of our knowledge, our work is the first review work to discuss the resource management issues for GPT-based model in cloud environment. The main **contributions** of this paper are as follows:

- We summarize the specific challenges in resource management for GPT-based model and provide a detailed description of these challenges.
- We present a comprehensive and general resource management framework for the GPT-based model that comprises seven different functional components.
- We propose and implement a machine learning (ML)-based method for resource profiling and prediction of LLM inference tasks, and validate its feasibility through experiments.

2 Challenges in Resource Management for GPT-based Model

By identifying the unique characteristics of resource management for GPT-based model, we summarized the specific challenges in resource management for GPT-based model deployed on clouds [5, 9, 15]. For a more visual representation, we presented these specific challenges in Figure 2. We noted some main challenges such as performance prediction and control, global manageability, resource heterogeneity, scalable resource management system, resource pricing strategies, model reliability, model parallelism and data parallelism, and we will discuss the details in the following sections.

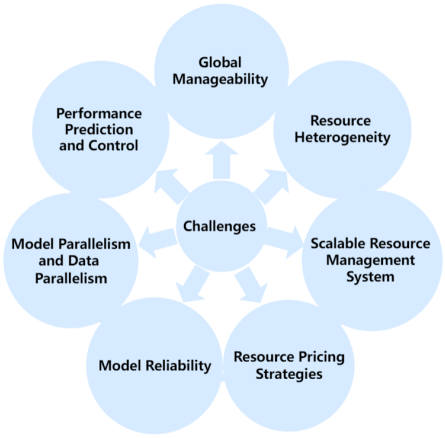


Fig. 2: Challenges in Resource Management for GPT-based Model

2.1 Performance Prediction and Control

The GPT-based model, constructed based on the Transformer [16] architecture, has a large parameter size and complex structure [12]. This complexity leads to significant computational requirements and the need for a large number of parameters for training and inference. Different tasks typically have varying complexities, resource demands, and performance expectations for the model. Even under the same workload and resource configuration, the performance of the model can be influenced by task characteristics and data properties. Additionally, different workloads and resource configurations can lead to variations in resource allocation, data parallelism, and other aspects that influence performance. These factors make it challenging to predict and control the behavior and performance of the model under different workloads and resource configurations.

2.2 Global Manageability

Global manageability refers to effectively managing and coordinating resources, including computational resources, storage resources, and network resources, in large and complex cloud environments. In the context of GPT-based model applications, challenges in achieving global manageability primarily manifest in the following aspects:

Resource scheduling and allocation: Given the massive computational, storage, and network resource requirements of the GPT-based model, efficient resource scheduling and allocation algorithms are needed. This includes dynamic resource allocation across different data centers and geographical locations to meet user demands and service-level agreements.

Resource monitoring and optimization: Achieving global manageability requires real-time monitoring of resource usage, performance metrics, and

health status, coupled with automated techniques for resource adjustments and optimization. Such monitoring mechanisms help maintain efficient resource utilization, ensure load balancing, and optimize performance bottlenecks, thereby enhancing overall system performance.

2.3 Resource Heterogeneity

Resource heterogeneity refers to the existence of various types or characteristics of resources within the same system or environment. These resources can include computational resources (such as CPUs, GPUs, and TPUs), storage resources (such as disks and solid-state drives), network resources (such as bandwidth and latency), and others. Resource heterogeneity implies differences in performance, scale, power consumption, and cost among these resources. For the GPT-based model, resource heterogeneity poses the following challenges:

Resource dependencies: It refers to the interdependencies and associations among different types of resources. In resource management, it is necessary to consider these dependencies and employ suitable resource allocation algorithms to optimize the synergistic effects among resources, thereby improving overall system performance. For example, in the GPT-based model, the provisioning of computational resources must match the capacity of storage resources and network bandwidth to ensure efficient data transmission and smooth model operation. By fully considering resource dependencies, resource allocation and utilization can be optimized, maximizing the system's potential.

Resource interoperability: Different types of resources are often provided by different vendors and technologies, necessitating addressing the challenge of resource interoperability. This involves establishing standards and protocols to ensure seamless integration and interaction among different types of resources, improving system compatibility and interoperability. Additionally, data and model transfer and sharing across different resources need to be addressed to enable collaborative work across resources.

2.4 Scalable Resource Management System

With the development of the GPT-based model, its scale has increased significantly, demanding huge computational resources. Moreover, as the GPT-based model is widely applied, data centers face concurrent requests and high throughput demands. Therefore, a highly scalable computing and storage infrastructure is required to support model execution and handle massive data. Furthermore, the resource management system must scale across multiple dimensions to handle resource management in large-scale data centers. The GPT-based model requires efficient management and allocation of computational, storage, and network resources, as well as task scheduling, to meet the training and inference requirements of the model. Additionally, the resource management system needs to dynamically allocate and flexibly expand resources to accommodate different application scenarios with varying scales and complexities. These requirements

pose important challenges to the resource management system for the GPT-based model.

2.5 Resource Pricing Strategies

Resource pricing strategies are crucial for the GPT-based model as they directly impact resource utilization, user satisfaction, and vendor profitability. However, several challenges exist in resource pricing strategies.

Firstly, accurately determining resource costs is a challenge. Resource costs are influenced by factors such as the vendor, geographical location, and usage volume. Therefore, a comprehensive consideration of these factors is required to ensure that resource pricing covers actual costs and attracts user adoption.

Secondly, balancing the supply-demand relationship is another challenge. Vendors aim to obtain revenues through resource sales while ensuring stable provisioning, while users seek resources at reasonable prices and sufficient support during peak demand. Therefore, resource pricing strategies need to strike a balance between supply and demand, meeting user needs while ensuring vendor profitability.

Additionally, achieving fair resource allocation and pricing is also a challenge. In multi-user environments, resources must be allocated on-demand to different users and priced based on usage. Given the variations in user demands and usage patterns, assuring fair resource allocation and pricing becomes a complicated problem.

2.6 Model Reliability

Due to the complexity of the GPT-based model, such as its large scale and long training processes, model failures during operation are inevitable. To ensure model reliability, systems must implement fault detection and fault tolerance mechanisms to handle resource failures or interruptions promptly. Fault detection mechanisms proactively identify potential system failures by monitoring model performance metrics, resource utilization, and other key parameters. Fault tolerance mechanisms include data backup and recovery strategies to ensure data integrity and service continuity. Data backup strategies involve regular backups of the GPT-based model parameters, training data, and other related data to ensure available backup data for recovery in case of failures. Recovery strategies ensure quick system recovery and maintain the continuity of user experience after failures or interruptions. Through these mechanisms, the system can rapidly detect and respond to faults, reducing the risks of system downtime and data loss, thus ensuring the reliability of the GPT-based model.

2.7 Model Parallelism and Data Parallelism

Parallelism mainly consist of model parallelism and dara parallelism. In model parallelism, challenges primarily include:

Model partitioning: The GPT-based model typically has a large scale, consisting of billions or even hundreds of billions of parameters. Partitioning such a massive GPT-based model into sub-models suitable for parallel processing is a challenge. Model partitioning needs to consider the dependencies within the model structure and the communication requirements among parameters to ensure correctness and efficiency in parallel computation.

Synchronization and communication overhead: Synchronization and communication are required among sub-models on different devices to ensure the transfer and aggregation of gradient information during training and enable effective parameter updates. Synchronization and communication operations can introduce additional computational and communication overhead, impacting the efficiency and performance of parallel computation.

Load balancing: Proper distribution of computational load is crucial for model parallelism to ensure balanced computation across devices. Load imbalance can result in computational resource waste and decreased efficiency.

In data parallelism, challenges mainly include:

Data partitioning and distribution: The GPT-based model has massive training data, and partitioning the data into multiple parts and distributing them to different devices is a challenge. Data partitioning needs to consider data balance and distribution efficiency to ensure the quality and performance of parallel training.

Data synchronization and consistency: In data parallel computation, model synchronization and consistency are crucial to ensure accurate parameter updates. Efficient data synchronization mechanisms are key to ensuring the effectiveness of parallel training.

Training speed limitations: In data parallel computation, the training speed may be limited by the slowest device. If some devices have slower computation speeds, it will affect the overall training efficiency and speed.

3 Resource Management Framework for GPT-based Model

In response to the specific challenges faced by the GPT-based model and based on the characteristics of resource management for GPT-based model, we propose a comprehensive resource management framework. This framework aims to effectively manage critical resources such as computational resources, storage resources, and network bandwidth required by the GPT-based model, thereby improving overall model efficiency and ensuring model reliability and service quality. Figure 3 demonstrates our resource management framework for the GPT-based model. The resource management framework is divided into several key components, including *Resource Monitor*, *GPT Task Scheduler*, *Resource Allocator*, *GPT Task Profiler*, *Synchronizer*, *QoS Manager*, and *Resource Adaptor*. The following will provide detailed introductions for each of these components.

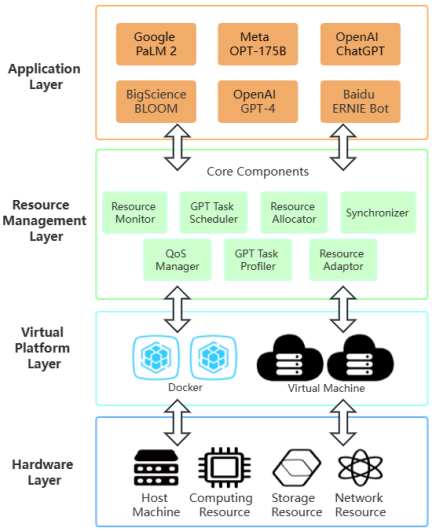


Fig. 3: Resource Management Framework for GPT-based Model

Resource Monitor: It is responsible for real-time monitoring of computational resources (e.g. CPU, GPU, memory), network resources (e.g. bandwidth utilization, network latency), and storage resources (e.g. disk) in the system. It collects and analyzes real-time resource usage and performance data, providing real-time feedback and reports to support task scheduling and resource allocation decisions. It also offers visual representations to show resource usage and performance metric trends.

GPT Task Scheduler: This component handles task scheduling based on requests from the GPT task queue. It considers task priority, resource requirements, timeliness, and related constraints to select suitable GPT-based model instances for task scheduling. By employing appropriate scheduling algorithms, it determines the execution order of tasks and assigns them to available GPT model instances.

Resource Allocator: It dynamically manages system resources based on task resource requirements, system resource availability, and load conditions. It employs intelligent resource allocation strategies to meet the execution needs of tasks. Additionally, the Resource Allocator may utilize resource prediction and load forecasting models to predict task resource requirements and system load conditions, enabling more accurate resource allocation and adjustments.

GPT Task Profiler: This component extracts and analyzes task attributes, resource requirements, and QoS needs of GPT tasks for enhanced understanding and handling. It identifies task types, input data features, and requirements, enabling customized parameter settings and providing essential references for data processing and answer generation. It also evaluates computational and storage

resources needed for accurate resource allocation and scheduling. Additionally, it identifies QoS requirements like response time, text generation speed, and accuracy, transferring them to the QoS Manager for effective evaluation and management.

Synchronizer: The Synchronizer facilitates efficient resource allocation and smooth execution of GPT tasks using distributed consistency protocols. It manages distributed locks to prevent resource conflicts, ensures consistent task states across nodes, and handles transactions to maintain atomic and consistent operations with rollback and recovery capabilities. Additionally, it manages fault recovery by detecting node failures and network issues, implementing measures like re-election or resource reallocation to ensure system stability and availability.

QoS Manager: It assesses and manages Quality of Service (QoS) requirements for GPT tasks. It evaluates and quantifies QoS needs such as response time, processing time, and accuracy from the GPT Task Profiler, devises optimization strategies based on these requirements and system resource constraints, and guides the GPT Task Scheduler and Resource Allocator. It also monitors real-time performance metrics, comparing them against QoS requirements, and initiates resource adjustments and optimizations to maintain desired QoS levels.

Resource Adaptor: This component is responsible for dynamic resource scaling based on system load, GPT task demands, and resource usage. It uses adaptive algorithms and prediction models to automatically adjust resource allocation for GPT tasks, achieving dynamic resource expansion and contraction. When the system load or task demands increase, the Resource Adaptor automatically scales up resources to meet the requirements and maintain system performance. Conversely, during low system load or reduced task demands, it timely scales down resources to reduce waste.

4 Task Profiling and Resource Prediction

As it is difficult to address all the aforementioned challenges in a single paper, in this work, we focus on addressing the challenges for GPT tasks profiling by precisely profiling the resources utilized by each task, which can be developed in the Task Profiler module in Figure 3. Our approach leverages K-Prototypes [1] clustering to categorize GPT tasks based on their resource consumption patterns. Based on clustering, we employ multiple machine learning algorithms for training and prediction. This method facilitates a detailed and accurate profiling of GPT tasks, thereby optimizing resource allocation and improving scheduling efficiency. This approach consists of 6 steps as shown in Fig 4 and details are as follows:

4.1 Task Design and Pre-profiling

Designing reasoning tasks is a critical step in ensuring the effectiveness and data diversity of profiling tasks. To fully demonstrate the performance of the ChatGLM2-6B model in answering various questions, we design questions based on their length, type, and language to ensure comprehensiveness and diversity.

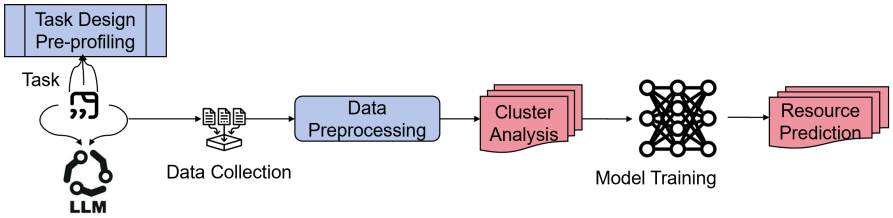


Fig. 4: Task profiling and resource prediction details

We created 182 questions ranging from 5 to 431 tokens in length[1]. The types of questions include text generation, summarization, translation, common sense Q&A, mathematics and logic, coding skills, and specialized fields. The questions are in both Chinese and English. Figure 5, Figure 6, and Figure 7 demonstrate the distribution of question types, languages, and lengths, respectively.

Through the question design, we can ensure the diversity and coverage of the selected questions, providing a more comprehensive and accurate analysis of GPT task resource usage. This detailed task profiling improves the model's overall performance across different tasks. Thus, our approach not only offers a fine-grained analysis of GPT task resource consumption but also ensures the comprehensiveness and representativeness of the analysis through diverse question design, providing a solid foundation for task resource profiling and prediction.

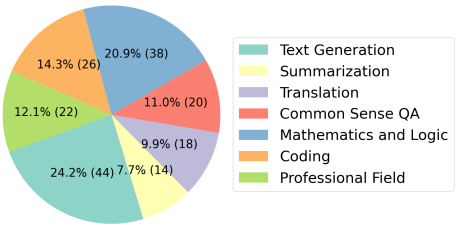


Fig. 5: Type distribution of questions

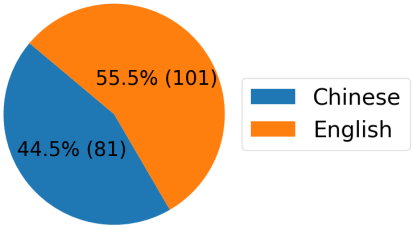


Fig. 6: Language distribution of questions

It is important to note that the length of a question is not determined by its text length but by converting the text of the question into a sequence of tokens (i.e., tokenization) and counting these tokens. Tokenization, the process of splitting continuous text into meaningful lexical units, is a crucial step in LLMs because the model needs to understand the basic units of the input text, known

[1] Detailed information of our designed questions can be found at: https://github.com/HYIUYOU/Resource-Management-for-GPT-based-Model-Deployed-on-Clouds

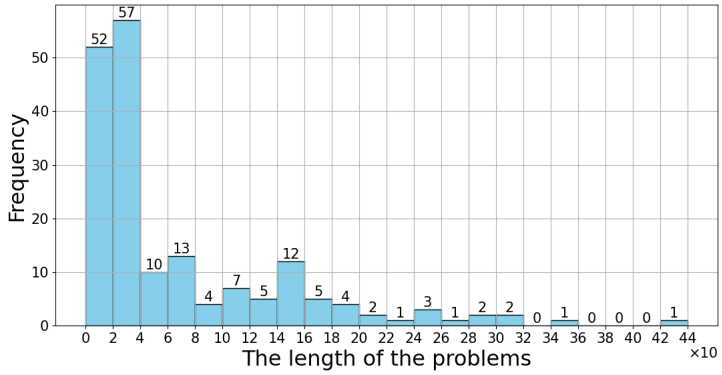


Fig. 7: Length distribution of questions

as tokens [14, 16]. Tokenization allows the model to better comprehend and process language. For instance, in text generation, the model generates coherent sentences or paragraphs by producing reasonable token combinations based on learned tokenization rules and language patterns. In LLMs, tokenization is performed by tokenizers, and different models have their specific tokenization methods. Using the number of tokens as a measure of question length aligns better with the model's inference mechanism, potentially allowing for more accurate resource usage prediction. After designing the questions, we store the length, type, and language data of each question in a CSV file for future use, facilitating preliminary analysis.

4.2 Data Collection

During data collection, two modules are utilized: one for model deployment providing inference services, and another for monitoring resource usage. Initially, the monitoring module executes the "nvidia-smi" command every 0.5 seconds to retrieve GPU utilization and memory usage, recording the data in a CSV file. Subsequently, the model deployment module loads the model, allowing real-time interactions. Pre-prepared questions are sent to the model, and responses are recorded. Each question is repeated three times to remove the randomness.

4.3 Data Preprocessing

During the data collection process, we record the changes in GPU utilization and RAM usage during the time of each task's inference phase. Additionally, to facilitate subsequent model training, it is necessary to extract key metrics from this data. To accomplish this, we wrote a Python program to process the data.

The data preprocessing workflow involves generating file paths with Python, reading and processing CSV data with Pandas toolkit, extracting and calculating resource usage metrics, storing these metrics in a list, creating a DataFrame, and saving it to a CSV file for further analysis. Through these steps, we successfully extracted a series of key metrics (e.g. maximum GPU utilization) from the original telemetry data, laying the foundation for subsequent analysis.

4.4 Cluster Analysis

The K-Prototypes clustering algorithm is designed for datasets that include both numeric and categorical features. It processes these different types of data by defining separate distance functions for numeric and categorical features. This algorithm integrates the concepts of K-Means (for numeric features) and K-Modes (for categorical features). The key idea is to partition the data points in a dataset into K clusters based on a certain similarity measure, maximizing the similarity between each data point and its respective cluster center, while minimizing the similarity between data points in different clusters.

The algorithm achieves its objective by minimizing the cost function $E(U, Q)$, where U is an assignment matrix with elements u_{ij} indicating the degree to which data point x_i belongs to cluster Q_j, and Q is the set of cluster centroids. The cost function is defined as the sum of the weighted distances from all data points to their nearest cluster centroid, as illustrated in Equation (1):

$$E(U, Q) = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij} d(x_i, Q_j), \tag{1}$$

where n represents the number of data points, k denotes the number of clusters, u_{ij} signifies the membership degree of data point x_i to cluster Q_j, and $d(x_i, Q_j)$ indicates the distance between data point x_i and cluster centroid Q_j.

The dissimilarity measure $d(x_i, Q_j)$ is computed based on the attributes of data point x_i and cluster centroid Q_j. For numerical attributes, the squared Euclidean distance is utilized, as described in Equation (2):

$$d(x_{il}, q_{jl}) = (x_{il} - q_{jl})^2, \tag{2}$$

where, x_{il} is the value of the l-th numerical attribute of data point x_i, and q_{jl} is the value of the l-th numerical attribute of cluster centroid Q_j.

For categorical attributes, a weighted simple matching distance is employed, as expressed in Equation (3):

$$d(x_{il}, q_{jl}) = \mu \cdot s(x_{il}, q_{jl}), \tag{3}$$

In this equation, μ represents the weight of the categorical attribute, and $s(x_{il}, q_{jl})$ is an indicator function that equals 0 when the attribute values x_{il} and q_{jl} are identical, and 1 otherwise.

These distance measures enable the algorithm to effectively compute the distances between data points and cluster centroids, thereby facilitating the clustering process.

The K-Prototypes clustering algorithm excels in handling mixed data types (numerical and categorical), offers flexibility in distance measures (e.g., Euclidean or Manhattan), and is computationally efficient, making it ideal for large datasets. Its interpretability also reveals inherent data structures and patterns.

In our clustering analysis, we integrate question attributes (length, type, language) with resource usage metrics (average and maximum GPU utilization,

RAM usage, total inference time). We apply the K-Prototypes algorithm to cluster inference tasks, followed by using a machine learning model (e.g., Random Forest) within each cluster to predict resource usage. This approach enhances prediction accuracy, generalizability, reduces modeling complexity, and mitigates noise sensitivity, better adapting to practical applications.

Using K-Prototypes, we have divided 182 data sets into 6 clusters. The number of samples in each cluster is shown in Figure 8, and the proportion of samples from each cluster within the total dataset is displayed in Figure 9.

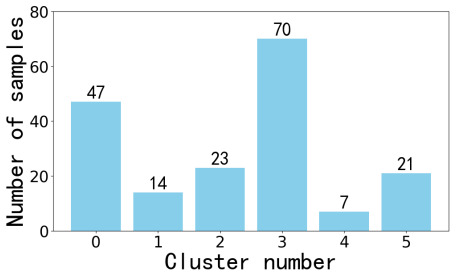


Fig. 8: Clustering results

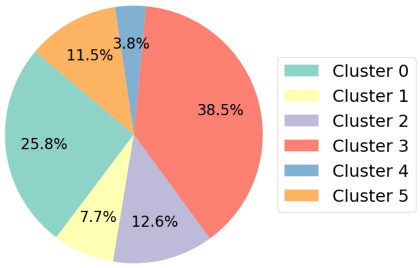


Fig. 9: Cluster sample number distribution

4.5 Model Training and Resource Prediction

We used three ML-based algorithms for training and prediction:

1. Random Forest (RF) [1]: It is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. The final prediction $\hat{y}$ is given by $\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x)$, where T is the number of trees, and $h_t(x)$ is the prediction of the t-th tree. It is particularly robust to overfitting due to the averaging of multiple trees, can handle high-dimensional data effectively, and is relatively easy to interpret.

2. Gradient Boosting (GB) [1]: It builds models sequentially, with each new model correcting the errors of the previous ones. The model updates are given by $F_m(x) = F_{m-1}(x) + \alpha \cdot h_m(x)$, where $F_m(x)$ is the boosted model at iteration m, α is the learning rate, and $h_m(x)$ is the new model fit to the residuals. This method is known for its high prediction accuracy, ability to handle complex data patterns, and flexibility in tuning through hyperparameters.

3. LightGBM (LGBM) [1]: It is an efficient implementation of the GB framework using tree-based learning algorithms. It is designed to be highly efficient and scalable for large datasets. The optimization objective can be formulated as minimize $\sum_{i=1}^{n} l(y_i, \hat{y}_i) + \Omega(T)$, where l is the loss function, y_i is the true value, $\hat{y}_i$ is the predicted value, and $\Omega(T)$ is the regularization term of the tree structure. LGBM's advantages include faster training speed, lower memory usage, and better accuracy due to its novel techniques like gradient-based one-side sampling and exclusive feature bundling.

In the model training and resource prediction phase, each sample within a cluster is split into a training set and a test set, with 70% of the data used for training and 30% for testing. We then perform individual predictions for four resource usage metrics: maximum GPU utilization, average GPU utilization, maximum Memory usage, and total inference time. For each test sample in the cluster, we calculate and record two metrics widely used: Relative Error (RE) and Mean Squared Error (MSE).

RE is defined as the proportion of the difference between the predicted ($\hat{y}$) and actual (y) values relative to the actual value, given by:

$$\text{RE} = \frac{|y - \hat{y}|}{|y|}. \tag{4}$$

MSE is the average of the squared differences between the predicted and actual values, formulated as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{5}$$

4.6 Evaluations

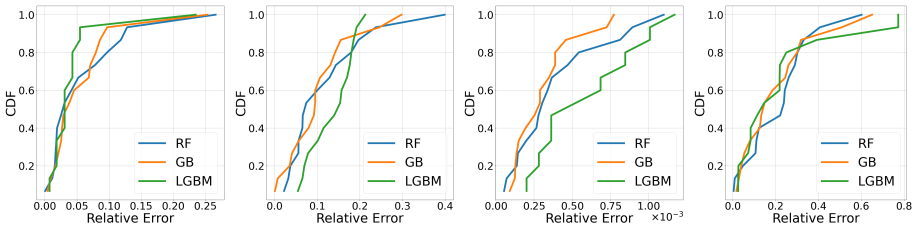
Given the diversity and data sufficiency of samples in Clusters 0, 3, and 5, we record two evaluation metrics (RE and MSE) for these clusters with three ML-based approaches as introduced in Section 4.5. To visually analyze the prediction results, we also plot the cumulative distribution function (CDF) of RE for test sample predictions in each cluster. In addition, we summarize the MSE of the predicted test samples in each cluster in Table 2.

Table 2: MSE for different ML-based approaches and clusters

Cluster	Max GPU utilization			Avg GPU utilization			Max memory used			Total inference time		
	GB	LGBM	RF	GB	LGBM	RF	GB	LGBM	RF	GB	LGBM	RF
0	32.94	**21.17**	40.07	**56.3**	73.51	73.56	**21.62**	68.52	38.58	4.97	**4.86**	5.25
3	**58.6**	261.09	78.47	**110.62**	297.51	114.18	64.92	88.33	**59.1**	12.63	19.82	**11.12**
5	5.47	**2.41**	5.59	**78.55**	114.4	91.87	11.62	21.57	**11.3**	**5.81**	8.21	6.03

The results are analyzed as follows:

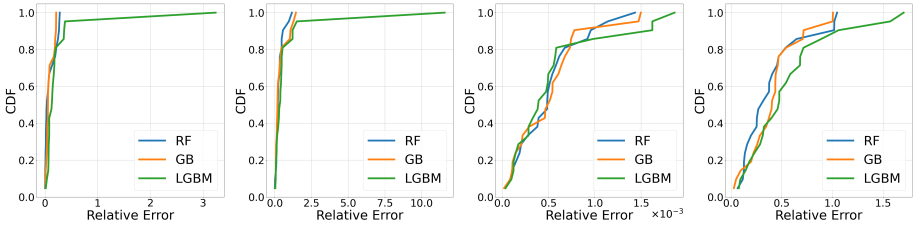
Cluster 0: Figure 10 illustrates the performance of various models. For maximum GPU utilization predictions, over 80% of the samples achieved a RE below 10%, with LGBM excelling at less than 5%. When predicting average GPU utilization, more than 80% of the predictions had a RE under 20%, with GB performing best at under 15%. In maximum memory usage predictions, all models achieve good performance, as every prediction had a RE below 0.12%, with GB leading at just 0.04%. Lastly, for total inference time predictions, the models performed acceptably, with over 80% of predictions within a 30% RE and similar performance across all models.



(a) (b) (c) (d)

Fig. 10: CDF comparison of RE for Cluster 0. (a) maximum GPU utilization. (b) average GPU utilization. (c) maximum memory used. (d) total inference time.

Cluster 3: As shown in Figure 11, over 80% of maximum GPU utilization predictions had a RE within 30%, with LGBM underperforming. For average GPU utilization, more than 80% of the predictions had a RE up to 100%, again with LGBM showing poor performance. However, in maximum Memory usage predictions, all models performed admirably, maintaining a RE under 0.75%. For Total Inference Time, the models' performance was acceptable, with both RF and GB maintaining a RE within 50%, while LGBM lagged at 60%.

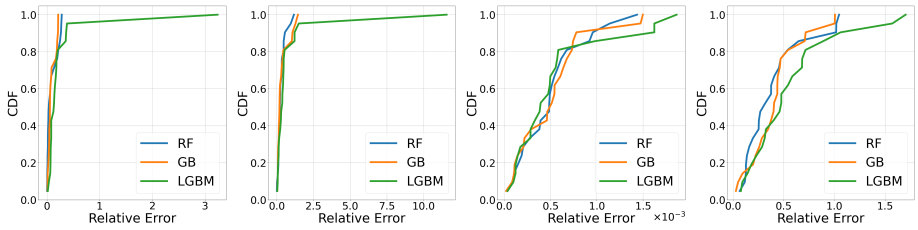


(a) (b) (c) (d)

Fig. 11: CDF comparison of RE for Cluster 3. (a) maximum GPU utilization. (b) average GPU utilization. (c) maximum memory used. (d) total inference time.

Cluster 5: According to Figure 12, approximately 80% of maximum GPU utilization predictions had a RE within 3%. In predicting average GPU utilization, over 80% of the samples maintained a RE below 20%, with all models showing similar performance. For maximum Memory usage, each model performed exceptionally well, with over 80% of predictions within a 0.05% RE, and GB and RF performing slightly better at 0.03%. Regarding Total Inference Time predictions, the models performed adequately, with about 80% of predictions within a 40% RE, where GB and LGBM showed superior performance.

To summarize, different models exhibited varying performance across different clusters and prediction metrics. Overall, the GB model demonstrated the best



(a) (b) (c) (d)

Fig. 12: CDF comparison of RE for Cluster 5. (a) maximum GPU utilization. (b) average GPU utilization. (c) maximum memory used. (d) total inference time.

and most stable performance. The RF model followed, showing stable results. The LGBM model, however, showed the least stability, particularly performing poorly in predicting GPU utilization in cluster 3. The results have shown the feasibility to use these approaches for task resource profiling.

5 Conclusions and Future Research Directions

In this paper, we introduced popular GPT-based models, identified unique characteristics of resource management for these models, and discussed evaluation metrics. We analyzed specific challenges and proposed a comprehensive resource management framework. In addition, we implemented and verified resource characterization and prediction methods for LLM tasks. Future research directions can be explored to highlight potential areas in resource management for GPT-based models. We summarize several future research directions for resource management for GPT-based model as follows:

Specialized Hardware for GPT-based Models: As GPT models scale up, their computational demands rise. Future research will focus on developing high-performance hardware. Manufacturers can create specialized AI chips (e.g., GPUs, FPGAs) optimized for GPT characteristics to support large-scale parallel computations.

Benchmarks for Performance Evaluation: Currently, there is a lack of standardized benchmarks for resource management for GPT-based model, and existing evaluation metrics are not comprehensive. Future efforts should focus on establishing more comprehensive test suites to evaluate resource management from multiple dimensions.

Resource Utilization Maximization: Future research needs to investigate more efficient resource management techniques to maximize the utilization of resources by the GPT-based model. Cloud data centers suffer from the issue in low resource utilization. This can be achieved through dynamic resource allocation, resource sharing, and parallel computing algorithms designed for GPT-based models.

Scheduling Algorithms and Metrics: Future research should develop specialized scheduling and optimization algorithms for GPT-based models, considering task priorities, resource constraints, and optimization metrics. These algorithms should aim to improve resource allocation and task scheduling. Evaluation metrics like throughput, response time, and task completion rate should be used to assess algorithm effectiveness. Additionally, new metrics suitable for GPT-based resource provisioning should be proposed.

Security Management: With the widespread use of GPT-based models, security concerns in resource management are rising. Future research should focus on: 1) *Data Privacy*: Protect user data during training and inference to prevent breaches. 2) *Model Security*: Safeguard model integrity and reliability from tampering and attacks. 3) *System Security*: Implement measures to protect computational resources from abuse, ensuring reliable resource management.

References

[1] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz. "A comparative analysis of gradient boosting algorithms". In: *Artificial Intelligence Review* 54 (2021), pp. 1937–1967.

[2] J. Bommarito, M. Bommarito, D. M. Katz, et al. "Gpt as knowledge worker: A zero-shot evaluation of (ai) cpa capabilities". In: *arXiv preprint arXiv:2301.04408* (2023).

[3] T. Brown, B. Mann, N. Ryder, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[4] W. Fedus, B. Zoph, and N. Shazeer. "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity". In: *The Journal of Machine Learning Research* 23.1 (2022), pp. 5232–5270.

[5] J. R. Gunasekaran, C. S. Mishra, P. Thinakaran, et al. "Implications of public cloud resource heterogeneity for inference serving". In: *Proceedings of the 2020 Sixth International Workshop on Serverless Computing*. 2020, pp. 7–12.

[6] Y. Jin, C.-F. Wu, D. Brooks, et al. "*S*3: Increasing GPU Utilization during Generative Inference for Higher Throughput". In: *Advances in Neural Information Processing Systems* 36 (2024).

[7] W. Kuang, B. Qian, Z. Li, et al. "FederatedScope-LLM: A Comprehensive Package for Fine-tuning Large Language Models in Federated Learning". In: *arXiv preprint arXiv:2309.00363* (2023).

[8] Z.-X. Li, A. Kunchukuttan, X. Zheng, et al. "vllm: A high-throughput framework for transformer-based language models". In: *Proceedings of the 2023 Conference of the Association for Computational Linguistics: System Demonstrations*. 2023, pp. 127–137.

[9] Z. Li, L. Zheng, Y. Zhong, et al. "AlpaServe: Statistical Multiplexing with Model Parallelism for Deep Learning Serving". In: *arXiv preprint arXiv:2302.11665* (2023).

[10] D. Narayanan, M. Shoeybi, J. Casper, et al. "Efficient large-scale language model training on GPU clusters using megatron-LM". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '21. St. Louis, Missouri: Association for Computing Machinery, 2021.

[11] H. Nori, N. King, S. M. McKinney, et al. "Capabilities of gpt-4 on medical challenge problems". In: *arXiv preprint arXiv:2303.13375* (2023).

[12] OpenAI. "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023).

[13] A. Radford, K. Narasimhan, T. Salimans, et al. "Improving language understanding by generative pre-training". In: (2018).

[14] R. Sennrich, B. Haddow, and A. Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1715–1725.

[15] S. Tuli, S. S. Gill, M. Xu, et al. "HUNTER: AI based holistic resource management for sustainable cloud computing". In: *Journal of Systems and Software* 184 (2022), p. 111124.

[16] A. Vaswani, N. Shazeer, N. Parmar, et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. Vol. 30. Curran Associates, Inc., 2017.

[17] L. Wang, L. Yang, Y. Yu, et al. "Morphling: Fast, Near-Optimal Auto-Configuration for Cloud-Native Model Serving". In: *Proceedings of the ACM Symposium on Cloud Computing*. SoCC '21. Seattle, WA, USA: Association for Computing Machinery, 2021, 639–653.

[18] B. Wu, Z. Zhang, Z. Bai, et al. "Transparent GPU Sharing in Container Clouds for Deep Learning Workloads". In: *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, 2023, pp. 69–85.

[19] M. Xu, W. Yin, D. Cai, et al. "A survey of resource-efficient llm and multimodal foundation models". In: *arXiv preprint arXiv:2401.08092* (2024).

[20] M. Xu, C. Song, H. Wu, et al. "esDNN: deep neural network based multivariate workload prediction in cloud computing environments". In: *ACM Transactions on Internet Technology (TOIT)* 22.3 (2022), pp. 1–24.

[21] D. Yin, Y. Liu, W. X. Zhao, et al. "A Survey of Resource-Efficient LLM and Multimodal Foundation Models". In: *arXiv preprint arXiv:2310.13165* (2023).

[22] C. Zhang, M. Yu, W. Wang, et al. "MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving". In: *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. Renton, WA: USENIX Association, 2019, pp. 1049–1062.

[23] W. X. Zhao, K. Zhou, J. Li, et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).