SMILE: A Cost-Effective System for Serving Massive Pretrained Language Models in The Cloud

Jue Wang
Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University
Hangzhou, China
zjuwangjue@zju.edu.cn

Ke Chen
Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University
Hangzhou, China
chenk@zju.edu.cn

Lidan Shou*
Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University
Hangzhou, China
should@zju.edu.cn

Dawei Jiang
Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University
Hangzhou, China
jiangdw@zju.edu.cn

Gang Chen
Key Lab of Intelligent Computing Based Big Data of Zhejiang Province, Zhejiang University
Hangzhou, China
cg@zju.edu.cn

ABSTRACT

Deep learning models, particularly pre-trained language models (PLMs), have become increasingly important for a variety of applications that require text/language processing. However, these models are resource-intensive and often require costly hardware such as dedicated GPU servers. In response to this issue, we present SMILE, a novel prototype system for efficient deployment and management of such models in the cloud. Our goal is to build a cloud platform from which tenants can easily derive their own custom models, and rent PLM processors to run inference services on these models at reduced costs. To facilitate this, we present a co-design of cost-effective storage and computation scheme for managing massive customized PLMs with constrained hardware resources via effective resource sharing and multiplexing. Our system consists of four core components: *vPLM creator*, *vPLM storage appliance*, *vPLM trainer*, and *vPLM processor*, which allow tenants to easily create, store, train, and use their customized PLM in the cloud without the need for dedicated hardware or maintenance. In particular, vPLM processors are virtualized from a physical machine, and are designed to have a multi-tenant nature, enabling efficient utilization of resources by precomputing the intermediate representation of PLMs and using adapters to provide customization instead of training the entire model. This allows tenants to host their PLMs in the cloud at minor costs. In our demonstration, we show that over 10,000 models can be hosted on one single machine without compromising the inference speed and accuracy. Overall, our system provides a convenient and cost-effective solution for tenants to host and manage PLMs in the cloud for their customized tasks.

CCS CONCEPTS

• **Computing methodologies** → *Natural language processing*; • **Information systems** → *Data management systems.*

KEYWORDS

Model Management, Model Deployment, Cloud Computing, Machine Learning

ACM Reference Format:
Jue Wang, Ke Chen, Lidan Shou, Dawei Jiang, and Gang Chen. 2023. SMILE: A Cost-Effective System for Serving Massive Pretrained Language Models in The Cloud. In *Companion of the 2023 International Conference on Management of Data (SIGMOD-Companion '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555041.3589720

1 INTRODUCTION

Pretrained language models (PLMs) [1, 2] have become increasingly popular in natural language processing tasks in recent years due to their strong performance and versatility. A typical PLM consists of stacked Transformer layers [5] trained on a large corpus of text data, which enables it to learn universal language representations and achieve impressive results on various downstream tasks. Despite the benefits, implementing and maintaining PLMs can be a daunting task, especially for smaller organizations or individuals without access to specialized hardware and expertise. Cloud computing offers a solution to the aforementioned PLM serving challenges, allowing PLMs to be hosted as a service [4, 7]. However, PLMs are usually large in model size and computationally intensive, making the traditional *per-tenant, per-model* serving strategy to be costly, particularly when each tenant needs his/her own customized model. For example, the base version of BERT contains 110M parameters, and thus only a few models can fit in a typical GPU. In order to serve a moderate number of tenants (<1000), the cloud platform

*Lidan Shou is the corresponding author.

SIGMOD-Companion '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 978-1-4503-9507-6/23/06…$15.00
https://doi.org/10.1145/3555041.3589720

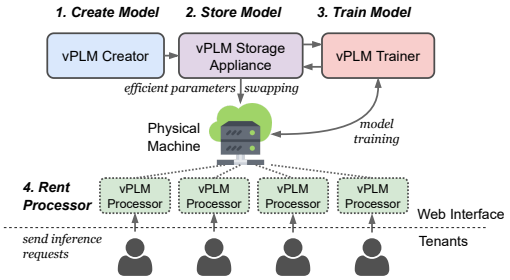


Figure 1: System architecture of the system.

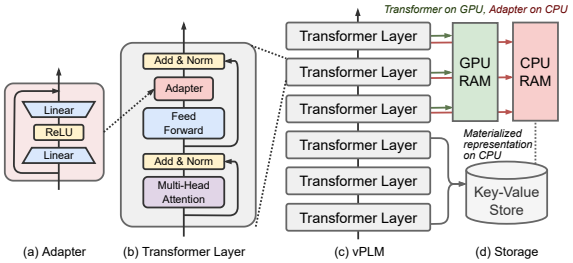


Figure 2: Illustration of a vPLM and its components.

may need to provision hundreds of dedicated GPUs, even if most tenants require limited workload.

To tackle the cost and scalability challenges, we introduce SMILE, a comprehensive PLM management system for cost-effective model creation, training, storage and hosting. Tenants can easily derive their custom models, and rent virtual PLM processors to run inference services on these models at reduced costs via effective resource sharing. To achieve this, we present a co-design of cost-effective storage and computation scheme for managing a large number of customized PLMs, referred to as 'vPLM', with constrained hardware resources. In particular, we show a single-GPU machine can support multiple model processors, each hosting a vPLM and serving its tenant. These model processors share physical hardware resources with necessary system optimizations to maximize hardware utilization. Meanwhile, each tenant has the impression of being served with a standalone vPLM without interference from others. We empirically demonstrate that one machine can be virtualized into more than 10K vPLM processors without significantly affecting the inference speed and accuracy.

Our system allows tenants to easily manage their customized PLMs in the cloud, without the need for purchase or maintenance of expensive hardware. We present the four core system components: 1) *vPLM creator* for model creation, 2) *vPLM storage appliance* for model storage with reduced size, 3) *vPLM trainer* for efficient model training, and 4) *vPLM processor* for high-throughput multi-tenant model inference. We present the benefits of using our service for hosting PLMs, including its ease of use, cost-effectiveness, and ability to handle a wide range of natural language processing tasks.

We show that our approach can significantly reduce the cost and resource requirements of serving PLMs, comparing to traditional methods with dedicated hardware. This makes it a viable option for organizations and individuals with limited budgets.

The rest of this paper is organized as follows. Section 2 presents the overview of SMILE, expatiating key components for model creation, training, storage, and inference. Section 3 presents the detailed demonstration plan.

2 SYSTEM OVERVIEW

This section presents an overview of SMILE. The key ideas can be summarized as follows: 1) *Space-Efficient Storage Scheme*: we use adapters [3], a compact neural network structure, to provides user customization, so as to reduce the per-tenant cost for model storage; 2) *High-Throughput Inference Scheme*: we materialize the lower layers of PLMs [6] to accelerate inference, and dynamically swap in the user-specific parameters to GPU on-demand to reduce the GPU memory requirements, thereby achieving high-throughput inference with a large number of PLMs on a single-GPU machine. 3) *End-to-End Model Management*: Our system offers a comprehensive solution for managing the entire process of creating, storing, training and deploying customized PLM with reduced cost.

Fig. 1 shows the system architecture. Our proposed system consists of a Web-based frontend and a backend using PyTorch to conduct training, management, and inference. Tenants can access the system via a Web browser to conveniently create, store, train, and use their customized PLMs in the cloud. Specifically, the system runs a server consisting of four core components:

- *vPLM Creator* helps tenants assemble models from PLMs provided by the cloud platform.
- *vPLM Storage Appliance* provides reliable storage for customized PLMs. Our system employs a space efficient strategy to to provide model customization, which can save tenants up to 99% in storage space.
- *vPLM Trainer* can be rented to train a PLM in a cost-effective way. vPLMs only require updates to the task-specific parameters, rather than the entire model, which allows for more efficient training at a lower cost.
- *vPLM Processor* is rented to process the inference from tenants. vPLM processors can be virtualized from the same physical machine. It is designed with a multi-tenant nature to support high-throughput and cost-effective inference.

2.1 System Components

In this subsection, we introduce the techniques used in the four key system components: vPLM creator, vPLM storage sppliance, vPLM trainer, and vPLM processor, as well as their implementation.

1) vPLM Creator. vPLM creator is designed to facilitate the creation of customized PLMs by tenants. Utilizing vPLM creator, tenants can select from a list of off-the-shelf PLMs, choose the version and configuration to create a customized PLM tailored to their specific needs. As presented in Fig. 2 (a-c), a vPLM consists of a backbone PLM and task-specific adapters:

Backbone PLMs. Backbone PLMs are pretrained on a large amount of data and are shared across tenants. In particular, we prepare several PLMs on specialized corpora to boost the performance on domain-specific tasks.

Adapters. vPLM creator uses adapters to provide customization. Adapters are small task-specific modules that can be inserted into a backbone PLM to adapt it to down-stream tasks. These adapters are fully-connected networks that adopt a 'bottleneck' architecture. Therefore, it is lightweight compared to PLM, accounting for only about 1% of the parameters of PLMs. Tenants can create a customized PLM by inserting adapters trained for public datasets. The tenants can also create randomly initialized adapters and train them using customized datasets later throught vPLM trainer.

The process of vPLM creator is made more efficient as tenants are only required to select and combine a limited number of PLMs and adapters, rather than designing the model structure and extensively training all parameters. This greatly simplifies the process of creating a customized PLM and allows tenants to effectively utilize PLMs for natural language processing tasks.

2) vPLM Storage Appliance. vPLM storage appliance is designed to provide secure and reliable storage for customized PLMs created by tenants. One of the key benefits of this service is its cost-effectiveness, as tenants are only required to store the task-specific parameters in their vPLM rather than the entire model itself. This approach reduces storage costs significantly as adapters, which adopt a bottleneck structure, typically have only a small fraction of the parameters of a full PLM, in some cases, up to 99% less.

Efficient Parameter Swapping On-Demand. The platform manages tenant-specific model parameters, i.e., adapters, in the host memory of the cloud server, and swaps them into GPU memory on demand during inference. We achieve an efficient implementation where the I/O time of loading these parameters can be properly hidden in GPU computation without increasing the inference time.

Materialization of Representations. Furthermore, to enable efficient inference, we also materialize the representations computed from the lower layers of PLMs. Specifically, we encode short text chunks with the lower layers of PLMs into independent representations and store them in the host memory handled by vPLM storage appliance. We implement an efficient index design to quickly look up these representations to approximate the actual input text representation, so as to save computation during inference. These materialized representations can be shared across tenants and thus the per-tenant storage cost can be negligible.

3) vPLM Trainer. vPLM trainer can be rented to help tenants train their customized PLMs efficiently. The training process for vPLMs is significantly faster than fine-tuning the entire PLM, and has reduced GPU memory footprint. This is due to the fact that vPLMs only require to update the parameters of the adapters, rather than the entire model. The speed-up in training time and reduction in GPU memory usage come from two main factors:

(a) The number of parameters that need to be updated during back-propagation is much smaller in vPLMs as compared to full PLMs. This means that the trainer only needs to compute the gradient of the parameters in the adapters and thus can be trained more quickly, leading to a 31% end-to-end speed-up.

(b) The GPU memory requirements are also reduced, since the trainer does not have to keep the optimizer states and gradients of the parameters of the backbone PLM (which are 3x the model size using Adam optimizer). Instead, only those for the adapters need to be kept. This leads to about 70% reduction in GPU memory usage.

Additionally, the platform can schedule their training during off-peak hours to further reduce the training cost. This enables tenants to fine-tune their models effectively while minimizing expenses.

4) vPLM Processor. Our system offers cost-effective model inference processing and allocates a vPLM processor for each model. It can virtualize a GPU into multiple vPLM processors to serve multiple tenants, thereby maximizing the hardware efficiency. The key techniques behind vPLM processors are presented as follows:

Virtualization. Tenants can rent vPLM processors virtualized from the same physical machine. This virtualization is non-trivial, as usually the inference requests that arrive at the same time correspond to different models, which can bring difficulties in maintain the hardware efficiency: (a) the physical machine may not have enough memory to host all models, and would therefore have to offload them to external storage e.g., SSD and load them back on demand. So the inference speed will be significantly limited by IO. (b) it can be difficult to batch these heterogeneous requests, therefore, in the worst case, the GPU can only run them one by one, leading to low GPU utility. However, thanks to the adapter technique, in our system, the physical machine can keep the backbone PLMs in the GPU memory and only needs to swap in the demanded adapters, only 1% of the parameters need to be swapped. This allows for more efficient virtualization and faster inference.

Pipelined Execution. The inference of a vPLM contains CPU, I/O, and GPU operations. Specifically, it includes loading the materialized representations, swapping the adapters into GPU, and Transformer computation. As a result, the sequential execution of these operations will result in a lot of 'bubble' – the processor either performs CPU, I/O, or GPU operations, while other parts are idle, leading to low hardware efficiency. In this demo, we propose to overlap the CPU, I/O and GPU operations with pipeline parallelism to improve the inference efficiency, which achieves a 25% speedup compared to naive sequential execution.

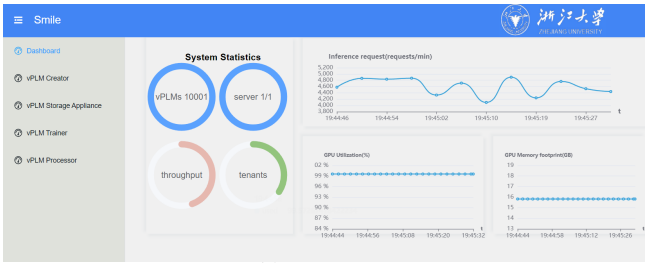
With the techniques listed above, we can use a single-GPU server to supply the needs of thousands of tenants.

3 DEMONSTRATION

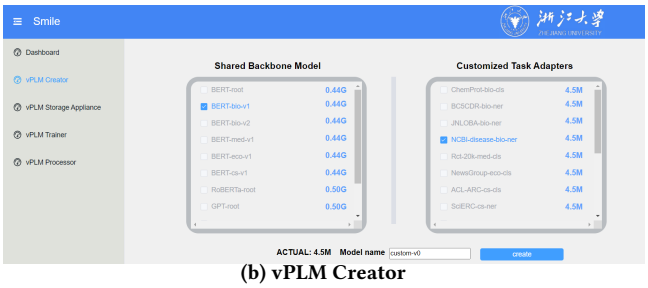
We demonstrate the main features of SMILE by serving 10K vPLMs on a GPU server. We invite the audience to interact with each major module of the system. The demonstration plan is as follows:

Environment. The demonstration system is hosted in a cloud server with 312GB RAM, one NVIDIA Quadro P5000 16GB, and Intel(R) Xeon(R) @ 2.40GHz with 64 cores. In this demo, we simulate 1,000 tenants, each creating 10 vPLMs and keeping sending an inference request every minute. The aggregated platform information is presented in Fig. 3a.

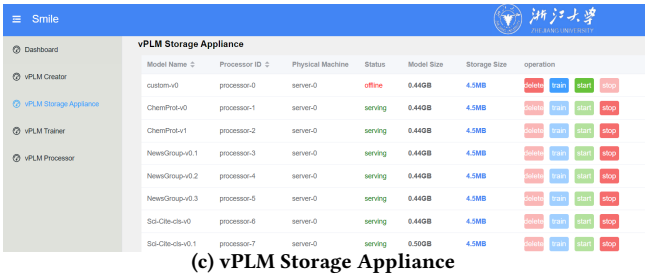
Model Creation. We demonstrate how to create a model using vPLM creator (Fig. 3b). The platform offers a variety of general-purpose PLMs as well as models that have been further pre-trained on domain-specific corpora. To enable customization, we allow tenants to choose from adapters that have been trained on public datasets, or to create new adapters. As an example, a tenant who wishes to create a text classification model for biological text can select the BERT-bio-v1 as the backbone model and the ChemProt-bio-cls adapter. In cases where none of the adapters are suitable for the tenant's needs, new adapters can be created and trained for



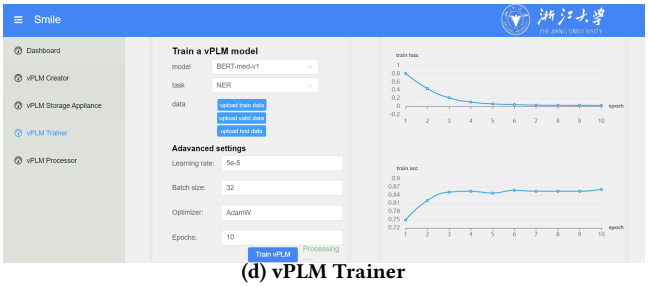
(a) Dashboard



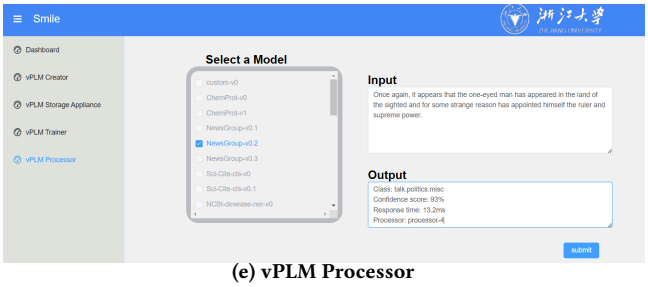
(b) vPLM Creator



(c) vPLM Storage Appliance



(d) vPLM Trainer



(e) vPLM Processor

Figure 3: User Interface of vPLM Creator, vPLM Storage Appliance, vPLM Trainer, and vPLM Processor.

later use. We stress that our vPLM creator allows tenants to create customized vPLMs in seconds.

Model Storage. Tenants rent vPLM storage to store the task-specific BERT models they created (Fig. 3d). In vPLM storage appliance interface, a tenant can view the overall status of all vPLMs he/she owns and manage individual models. Here, we demonstrate that the tenant only needs to store adapters rather than the entire BERT model. As a result, the price paid for model storage is significantly lower, making it a highly cost-effective service. Specifically, we here show that the entire model size is hundreds of MBs while the actual storage consumption is ~4 MB only.

Model Training. After creating a vPLM with randomly initialized adapters, the tenant can upload his/her customized datasets to the platform to fine-tune the model (Fig. 3c). The tenant can specify the hyperparameters for training, including the learning rate, batch size, optimizer type, and training epochs, etc. Training will be arranged in the background in off-peak hours. The metrics including the loss and accuracy in each iteration will be displayed in curves to illustrate the training progress. We highlight that the training time of our approach is significantly faster than training the entire PLM for achieving the same prediction accuracy.

Model Inference. Our system allocates a vPLM processor for each model. To perform inference, as shown in Fig. 3e, a tenant can select a vPLM he/she owns, e.g. NewsGroup-v0.3, a classification model for newsgroup data. The tenant can type input text in the 'Input' text box, and the result will be displayed in the 'Output' text box along with detailed execution information. Tenants can also use an HTTP based API to integrate the service into their own applications. We demonstrate that we can use a single-GPU machine to serve a large number of vPLMs. The backbone PLM preserves in the GPU while the adapters are loaded to GPU on demand. Thus, the price paid for each vPLM processor is low, since the GPU's operational cost is amortized among all loaded vPLMs. Overall, our approach is highly cost effective compared to conventional model deployment strategies with dedicated hardware for each model.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China (2022YFB3304100).

REFERENCES

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT*.

[3] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proc. of ICML*.

[4] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. 2015. Mlaas: Machine learning as a service. In *Proc. of ICMLA*.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS* 30 (2017).

[6] Jue Wang, Ke Chen, Gang Chen, Lidan Shou, and Julian McAuley. 2022. SkipBERT: Efficient Inference with Shallow Layer Skipping. In *Proc. of ACL*.

[7] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2017. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proc. of Internet Measurement Conference*.