Computing in the Era of Large Generative Models: From Cloud-Native to AI-Native

Yao Lu[1,2], Song Bian[3], Lequn Chen[4], Yongjun He[5], Yulong Hui[6], Matthew Lentz[13], Beibin Li[19], Fei Liu[18], Jialin Li[2], Qi Liu[7], Rui Liu[8], Xiaoxuan Liu[9], Lin Ma[10], Kexin Rong[11], Jianguo Wang[12], Yingjun Wu[17], Yongji Wu[13], Huanchen Zhang[6], Minjia Zhang[14], Qizhen Zhang[15], Tianyi Zhou[16], Danyang Zhuo[13]

Nativ[1], National University of Singapore[2], University of Wisconsin, Madison[3], University of Washington[4], ETH Zürich[5], Tsinghua University[6], University of Hong Kong[7], University of Chicago[8], University of California, Berkeley[9], University of Michigan[10], Georgia Institute of Technology[11], Purdue University[12], Duke University[13], University of Illinois, Urbana-Champaign[14], University of Toronto[15], University of Maryland[16], RisingWave[17], ByteDance[18], Microsoft[19]

ABSTRACT

In this paper, we investigate the intersection of large generative AI models and cloud-native computing architectures. Recent large models such as ChatGPT, while revolutionary in their capabilities, face challenges like escalating costs and demand for high-end GPUs. Drawing analogies between large-model-as-a-service (LMaaS) and cloud database-as-a-service (DBaaS), we describe an AI-native computing paradigm that harnesses the power of both cloud-native technologies (e.g., multi-tenancy and serverless computing) and advanced machine learning runtime (e.g., batched LoRA inference). These joint efforts aim to optimize costs-of-goods-sold (COGS) and improve resource accessibility. The journey of merging these two domains is just at the beginning and we hope to stimulate future research and development in this area.

1 INTRODUCTION

Recent advancements have witnessed the advent of generative AI, notably in the realm of stable diffusion models such as SDXL and large language models such as ChatGPT. These models have shown revolutionary abilities in understanding human language and generating realistic content. As these models evolve and increase in size and complexity, their applications expand across a wide range of domains, from advanced chatbots and virtual assistants to data analysis, forecasting, and even creative endeavors like writing, art, and music composition.

However, the proliferation of these models and their applications has brought about unique challenges to the systems and infrastructures underneath, both vertically in terms of increased individual model and task complexity, and horizontally with the exploding number of users and applications. Among many practical concerns, *cost of goods sold (COGS)* is one of the major barriers to pushing these models further in AI applications. Take, for instance, OpenAI's GPT-4. The model inference is priced at $0.12 per 1K tokens[1], which is more than the expected revenue per usage in many web-based applications such as search engines [2]. The business justification for using these large models comes under scrutiny. To train these models often at the scale of tens to hundreds of billions of parameters, the vast amounts of data required, coupled with the computational power needed, resulted in significant expenses. The iterative nature of model refinement also piles up the costs over time. Besides COGS, the *accessability* of GPUs, essential for training and serving large models, has also emerged as a bottleneck. The surging demand for large models has intensified the competition for high-end GPUs; many research institutions and small enterprises grapple with limited access to these specialized hardware resources.

In comparison, models such as Meta's Llama-2 [75], offer a notable financial benefit due to their size; being smaller (a few to tens of billions of parameters), these models are inherently cheaper to train, use, and maintain. Instead of aiming for broad capabilities as seen in gigantic generic AI models, these models are often fine-tuned for specific tasks. This specialization ensures optimized performance without the overhead of unnecessary model abilities [5, 35, 81]. The large number of specialized models adds to the complexity of the systems underneath.

The current AI software stack features a modular architecture. Model runtime frameworks such as TensorFlow [4] and PyTorch [61] are the key enablers for upper-layer systems such as Huggingface Transformers [82], Megatron-LM [56, 71] and DeepSpeed [65, 66, 72], amplifying the model training and deployment efficacy in distributed settings. However, current efforts predominantly concentrate on vertical scalability and efficiency in single-model systems. The challenges associated with resource accessibility and optimizing COGS in extensive *scale-out configurations* have yet to become research and development focuses.

Looking back: cloud-native technologies. Arguably, large generative models, at a high level, function analogously to databases. They capture data knowledge and, upon receiving a query or prompt, assemble and provide a relevant response. To enhance the COGS and resource accessability for large generative models, it is instructive to revisit the evolution of the cloud as well as database-as-a-service (DBaaS) over recent years. Importantly, the challenges faced are not unprecedented; many fall under the purview of cloud-native computing, a paradigm shift that has redefined our understanding of the cloud ecosystem.

Specifically, by utilizing containers and orchestrators like Kubernetes [42], cloud-native computing ensures scalability, resilience, and modularity. The microservices structure decomposes systems into distinct modules, enhancing agility and simplifying maintenance. Multi-tenancy permits multiple systems and tasks to share common infrastructures, thus optimizing resources and cost while ensuring isolation. With auto-scaling and serverless computing,

[1]With 32K context. Base model inference price in October 2023.

the cloud-native architecture significantly improves COGS when usage patterns change [40, 41]. In this vision paper, we unravel the nuances of how systems for large generative models can be seamlessly woven into the cloud-native computing architecture.

AI-native: new challenges and opportunities. Inherit from the legacy cloud-native systems, several of the aforementioned technologies can be effectively applied, including the containerization of machine learning runtime and the dynamic scaling of model inference tasks. It is worth noting that, in these scenarios, machine learning operations are often treated as black boxes. By fostering a deeper integration and potentially co-designing machine learning runtime with cloud-native systems, we pave the way for the emergence of a novel *AI-native* computing paradigm.

At the core of the AI-native paradigm lies the process of training, fine-tuning, and deploying large models, with goals remaining unchanged for improved COGS and resource accessability. Take the following use case for instance of AI-native computing: as a majority of end users cannot afford to train foundational models, they fine-tune open-source models with their specific data and applications. Currently serving hundreds of models at the same time can be of low efficiency, even if these models are fine-tuned from the same foundational model. In fact, this is a typical multi-tenancy scenario. We note a prior work, Punica [20], that developed an ML runtime with a batched LoRA inference mechanism which improved the output throughput by up to 14x. This is an exemplar of the AI-native computing paradigm.

To alleviate the resource accessibility issue, we may turn to serverless computing and more versatile cloud infrastructures from emerging decentralized GPU providers such as Vast.ai and Akash Networks who offer cost-effective GPU containers. However, hosting diverse large-model jobs and building, operating, and optimizing systems upon such heterogeneous and sometimes geo-distributed infrastructures create unforeseen challenges.

Nevertheless, COGS and resource accessability are merely the tip of the iceberg for this AI-native paradigm. Equally vital topics include serving broader AI applications and so on; we defer more discussions to Section 4. These challenges highlight the complexities involved in the deep integration of cloud-native methodologies with large generative models and suggest numerous exciting opportunities for future research and development.

Key takeaways of this paper can be summarized as follows:

- The current AI software stack focuses on vertical scalability and efficiency in single-model systems; horizontal scaling-out scenarios have yet to become research and production focuses.
- Large-model-as-a-service (LMaaS) and database-as-a-service (DBaaS) have commonalities. Many cloud-native designs are readily applicable to improve system efficiency and other aspects. Meanwhile, many other techniques require further adaptations and co-design between ML and cloud systems. This formulates a novel AI-native computing paradigm, and we provide a tentative list of interesting topics
- Many open questions remain. We provide an outlook on the research and production challenges and opportunities.

2 BACKGROUND

Cloud-native computing has redefined software development, leveraging concepts such as containerization and microservices for scalability and adaptability. Classic software like legacy customer relationship management (CRM) platforms and database systems have been limited by the constraints of proprietary infrastructures. One cornerstone of the transformation towards cloud-native computing is containerization which allows developers to wrap applications in containers, ensuring uniform behavior in heterogeneous environments. Kubernetes [42] and other container orchestration tools take this a step further. Say an e-commerce platform witnesses a sudden surge in traffic during a sale, an orchestrator can dynamically scale resources, ensuring that the website remains responsive.

The microservice architecture decomposes applications into discrete, function-oriented services. Cloud-native platforms like YouTube exemplify this strategy. Instead of a monolithic structure, they operate via microservices, each dedicated to specific tasks, ranging from user authentication and data storage to video encoding and streaming [28]. This approach offers several advantages. First, it enhances robustness; isolated failures in one service will not disrupt others. Second, maintenance becomes more manageable; individual services can be updated or debugged without affecting the entire ecosystem. Last, this architecture improves agility. Developers can concurrently innovate and roll out new features.

It is worth noting that multi-tenancy plays an important role in cloud-native computing. Rooted from the shared-everything architecture in the 80s, software platforms started to harness multi-tenancy to serve multiple users with shared resources on the set of infrastructures, instead of spinning up individual instances for each user [29, 36, 49, 55]. This greatly optimizes resource usage and reduces user costs. Amongst many successful cloud-native systems, database-as-a-service (DBaaS) exploits multi-tenancy from various aspects including infrastructures, datasets, queries, and relational operators. Prominent industry players like Snowflake [22] and Amazon Redshift [34] have emerged, driven by the imperative to provide enhanced services at competitive price points.

Large-model-as-a-service (LMaaS), fundamentally a cloud-based approach, bridges the gap between advanced AI and real-world applications by offering efficient, scalable, and accessible deployment mechanisms. Organizations, especially those without vast computational resources, can access state-of-the-art models without the overhead of training, maintaining, and deploying the models. By abstracting the complexities from the systems and infrastructures aspects, LMaaS offers simple interfaces to end users:

Model training. $\mathcal{W}_m \leftarrow \texttt{train}(m, \mathcal{D}, C)$ builds the model m on dataset $\mathcal{D}$ with hyper-parameters C; the output is the weight (i.e., model parameters) of the model $\mathcal{W}_m$.

Model fine-tuning. $\mathcal{W}'_m \leftarrow \texttt{finetune}(m, \mathcal{W}_m, \mathcal{D}', C')$ is a special case of model training to gain domain-specific knowledge at a much smaller cost. The input includes the weights $\mathcal{W}_m$ of the previously-trained model. Different training hyper-parameters C' are used with a dataset $\mathcal{D}'$ that is also often smaller. In a common case, the disparity in C' includes a much smaller number of epochs while most other knobs remain unchanged.

Model inference. $r \leftarrow \texttt{inference}(m, \mathcal{W}_m, q)$ generates content r with a trained model and an input query q.

Recently, Low Rank Adaptation (LoRA) [38], a fine-tuning strategy to further reduce computational cost, has attracted both research and production attention. Such a strategy only slightly changes the user contract: for model fine-tuning, the contract changes to $\Delta m, \mathcal{W}_{\Delta m} \leftarrow \texttt{finetune}(m, \mathcal{W}_m, \mathcal{D}', C'')$. The key idea is to keep the weights of the base model $\mathcal{W}_m$ unchanged while training a small *LoRA adaptor* $\mathcal{W}_{\Delta m}$ that has only small fractional trainable parameters. As a result, the fine-tuning speed is faster by orders of magnitude. Therefore, the new model inference contract is $r \leftarrow \texttt{inference}([m, \Delta m], [\mathcal{W}_m, \mathcal{W}_{\Delta m}], q)$ where the input query q goes through both the base and the adaptor models; doing so incurs little impact on the inference latency since the adaptor model is small.

LoRA provides an agile and low-cost contract in addition to the simple LMaaS interfaces. The base model can be viewed as a public asset for many different users and application scenarios, while the adaptors serve as specifications. We will show how this new contract plays an important role in cloud-native, AI-native computing for the remainder of this paper.

HuggingFace and similar marketspaces are yet another driving horse of LMaaS that provides the community with state-of-the-art pre-trained models, finetuned adaptors, datasets, and a platform for training and deploying custom models. Thousands of established organizations publish models and other resources on their platform which no doubt pushes the entire ecosystem of LMaaS further.

3 CLOUD-NATIVE FOR LARGE MODELS

The intersection of LMaaS and cloud-native computing is not merely a convergence of two technological domains; it is a fusion that has the potential to reshape the very fabric of modern cloud and AI. The synergies can be described below.

Containerization. It is worth noting that recent advancements in virtualization technologies enable a transparent pass-through to low-level operating systems. On the AI and deep learning framework level, popular offerings such as PyTorch and TensorFlow have integrated the runtime for heterogeneous compute architectures such as CPU, GPU and FPGA. These developments, as well as the protocol design for collaborative work in-between various container instances, facilitate effortless delivery of the software stack for large models in dynamic deployment scenarios.

Orchestration plays a critical role in ensuring elasticity, scalability, and resilience, which are fundamental for LMaaS to deliver robust and economically viable pay-as-use services. Given the fluctuating nature of user workloads and the underlying infrastructure dynamics, it is imperative to draw from, and further refine, the best practices inherent in current cloud-native solutions. Take, for instance, the changing query patterns: as they shift from predictable or sporadic requests to intense traffic surges, serverless technologies are equipped to dynamically and quickly scale resources upwards and downwards. They can also facilitate the migration of containers tasked with model-serving functionalities. Such adaptability minimizes resource redundancy and thus improves the COGS. Yet another dimension to consider is the disparate resource availability and heterogeneity across various segments of the cloud. Ensuring efficient orchestration in such a diverse environment demands meticulous resource management strategies involving a mixture of resource allocation, queuing, and adjustments on-the-fly, especially when catering to large-scale, distributed ML workloads.

Microservices. Beyond the fundamental services previously mentioned, there is a clear shift towards integrating large models with enhanced functionalities and additional data. As the machine learning lifecycle matures, it increasingly intersects with sophisticated algorithms and methodologies. A case in point is the recent adoption of multi-stage training strategies in large language models, particularly the use of Reinforcement Learning with Human Feedback (RLHF) [58]. Concurrently, the advent of diverse AI applications, including search engines, speech synthesis, business data analytics, and image generation, demands processing capabilities for multi-modal data. Given these complexities, cloud providers are compelled to architect and streamline microservices and augment the models backed by disaggregated, large-scale data stores. This is essential to accommodate the rapid-growing diversity and volume of data and the evolving landscape of machine learning algorithms.

Multi-tenancy, a design principle that affects each of the technologies above, can be viewed from different aspects. From the model perspective, there can be multiple base models and each company with multiple LoRA adaptor models; there can also be concurrent input queries from different users. From the task perspective, there can simultaneously be training, fine-tuning, serving, data preparation, management, and operational workloads. In these scenarios, it is essential to consolidate the computation and save GPU memory. These LM use cases also open up opportunities for multi-tenant systems and infrastructures underneath to further save costs and improve efficiency, for example, aggregated network communication and shared data infrastructures in distributed training and dataset preparation tasks.

Armed with these capabilities, we are poised to establish cloud-native infrastructures tailored for large generative models. In the subsequent sections, we will describe our initial endeavors within select areas of the aforementioned research spectrum.

Example: RAG-as-a-Service vs BI-as-a-Service. Retrieval Augmented Generation (RAG) [45] has become an important topic in natural language processing. It extends the capabilities of large language models by incorporating an information retrieval system for reference data. RAG is versatile in diverse applications such as document analysis, summarization, personal chatbot interactions, and code comprehension.

Figure 1 (Right) illustrates the workflow and architecture of RAG-as-a-service (RAGaaS), compared with a typical cloud-native Buiness-Intelligence-as-a-Service (BIaaS) [18] architecture in Figure 1 (left). In the right figure, the process begins with the extraction of text documents or code from clients, followed by segmenting inputs into smaller units. These segments are then transformed into vectors and stored in a dedicated vector storage system. When a user submits a query, the system retrieves the most relevant data chunks from the vector storage. The user's query and the retrieved data are then combined and passed into the LM inference model to generate an answer. In comparison, BIaaS in contemporary cloud-native

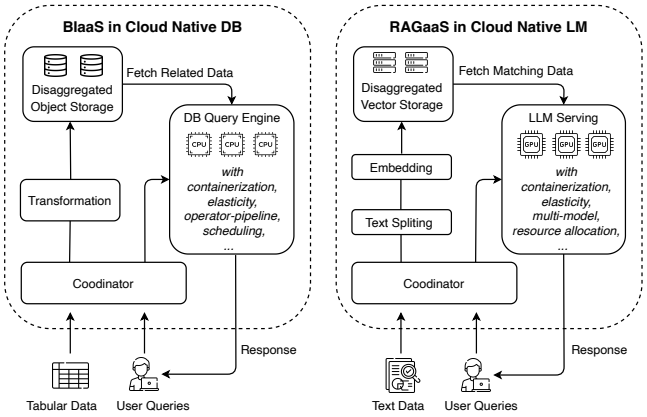
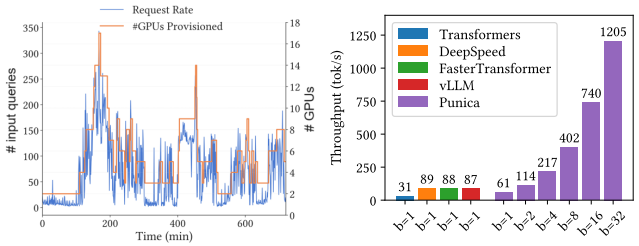


Figure 1: BIaaS and RAGaaS share architectural commonalities.



(a) Elastic resource scheduling. **(b) Batched LoRA inference.**

Figure 2: Preliminary results to verify AI-native computing.

database systems exhibits a similar workflow. It also involves data processing stages for extracting, splitting, embedding, and loading tabular input data. SQL queries initiate data retrieval and execution of corresponding database operations, mirroring the data retrieval and LM inference processes in RAG.

Adopting cloud-native techniques. When bridging the concepts of database and large model applications, some cloud-native technologies can be readily applied in the context of RAGaaS. For instance, large model inference endpoints can be containerized for streamlined deployment; functionalities like vector database search can be offered as microservices. A cloud orchestrator improves system efficiency by leveraging disaggregated storage and dynamically scaling containers and GPUs resources. To enhance multi-tenant RAG workload processing, implementing query optimization and batching techniques have proven to be effective, enabling batched concurrent queries and routing different queries to various large models in accordance with user service level agreements.

While cloud-native technologies offer immediate benefits, treating large model tasks and system components as black boxes, there remains a wealth of untapped potential for specialized large-model optimizations.

4 PRELIMINARY AI-NATIVE COMPUTING

In this section, we explore the prospects arising from the deep integration of large models with cloud-native technologies, paving the way for a novel AI-native computing paradigm. To achieve so, we mention some prior work as well as some of our preliminary experiments. Our purpose here is not for completeness but to point out the potential values for future research in this area.

Case Study 1: Elasticity. The first idea from legacy cloud-native computing that we try out is the elastic resource scheduling for serving 7B llama-2 models on a cluster of NVidia 4090 GPUs. This is of practical importance to improve resource utilization when incoming query patterns vary. Our cloud orchestrator collects runtime telemetries and applies a heuristics-based feedback mechanism to scale up/down the number of inference instances. In our preliminary experiments, we synthesize incoming queries with a realistic whole-day pattern and demonstrate our findings in Figure 2a. Surprisingly, with simple scaling heuristics, our system already provides prompt scaling; models can be loaded quickly in seconds, with the aid of high-speed networking and PCI-E 5.0 bus. This indicates that elasticity is possible and beneficial in the new AI-native computing paradigm to improve resource usage efficiency.

Case Study 2: Multi-tenancy. Here we elaborate more on the example shown in the earlier sections. When many LoRA fine-tuned models [38] share the same foundational model, a novel machine learning runtime, Punica [20], allows batching inference jobs from multiple user queries. This is a typical multi-tenant scenario in classic cloud systems where users share the same software infrastructures. The delta here is the model inference procedure which involves one single copy of the base model and multiple LoRA adaptors. Punica implement a LoRA batching mechanism with a custom CUDA kernel which enables fast communication between the base model and adaptors. Figure 2b compares Punica with state-of-the-art large model serving systems, with b = batch size on the x-axis, and baseline systems such as Huggingface Transformers [82], DeepSpeed [66], and vLLM [43] running at batch size 1, since they do not support batched LoRA inference. Punica delivers 14x throughput compared to vLLM at a batch size of up to 32 with 7B llama-2 base model on a single A100 GPU with 80G memory.

Case Study 3: Hybrid cloud deployment and optimization. As the cloud paradigm expands and becomes increasingly diverse, notably by the ubiquitous smart devices and the Internet of Things. A prior work, JellyBean [84], introduced a system that deploys and optimizes machine learning inference workloads across such heterogeneous infrastructures.

JellyBean has the following cloud- and AI-native designs: (1) containerized runtime for deploying ML inference jobs on heterogeneous infrastructures such as IoT devices and cloud data centers, and (2) a hybrid cloud orchestrator that imploys query optimization techniques to choose the most cost-effective models aligned with specified service-level objectives (like throughput or accuracy) and assigns workers across different segments of the cloud. With these designs, JellyBean empirically reduces the total serving cost by up to 58% for tasks like visual question answering and by up to 36% for vehicle tracking from the NVIDIA AI City Challenge, compared to prior solutions for hybrid-cloud ML deployment.

Remark. Through three case studies, the advantages of a cloud- and AI-native design in enhancing the efficiency and mitigating the costs of large models become evident. While our investigations remain preliminary and address only specific aspects, we anticipate comprehensive and in-depth studies to follow, given the rising significance of large generative models in research and production.

5 OUTLOOK TO THE FUTURE

While the previous section exemplifies a few valuable directions in the AI-native computing paradigm, there are more open challenges and opportunities that are equally important. We elaborate on some of them in this section.

Runtime for large generative models. While this vision paper mostly discusses scaling large generative models horizontally, vertical scalability and efficiency of the machine learning runtime are still of critical importance. Large models require substantial memory that often exceeds the GPU capacity; key-value caches used in the transformers push this further. Transferring data between High-Bandwidth Memory (HBM) and Static Random-Access Memory (SRAM) consumes memory bandwidth, hence sparse computations make inference bandwidth-bound and result in underutilized GPU compute capacities [70]. Building an efficient inference engine that fully leverages the available memory and computation remains a prevailing challenge.

Recent advancements in machine learning have introduced the concept of Mixture of Experts (MoE) models [27, 68] and speculative decoding [19, 44] to enhance the model's efficiency and quality. The primary idea behind these innovations is to curate a collection of diverse, smaller models, each tailored for specific tasks, thus avoiding the exhaustive use of complete models and conserving computational resources. For example, models like Llama-2 already exhibit varying sizes, making them more cost-effective when used strategically. However, deciding if the potential savings and increased efficiency outweigh the initial overhead of creating and maintaining multiple models is non-trivial. Insights from database management systems, such as materialized views [30], can be applied to efficiently handle recurring queries without repetitively calling large models.

As spot innovations continue to advance rapidly, we anticipate that significant breakthroughs will emerge from both the systems and machine learning domains. Examples include FlashAttention [23, 24] and PagedAttention [43] mechanism which incorporates ideas from both domains; these innovations will push current solutions to the next level.

Continuous learning and serving systems. In many applications, data comes in real-time. To provide users with useful and up-to-date insights, serving systems may be required to keep learning from data streams. Traditional approaches of periodically retraining or updating large models prove to be computationally expensive and time-prohibitive [46].

Systems that learn continuously are the solution to keep the models afresh. Classic stream processing systems like Naiad [52], Spark Streaming [77, 87], Apache Flink [17], and RisingWave can be adapted to process continuous data streams in (near) real-time. Machine learning and data management technologies such as LoRA fine-tuning [38] and indexing can be used to process new data and update existing models efficiently. Machine unlearning [15], i.e., forgetting specific knowledge, is notable in future research and production challenges to remove data that is not needed.

Real-time serving of large models presents challenges, including scenarios where queries arrive in parallel but are not synchronized [33, 64, 69]. Prior research such as Orca [85] has provided tentative solutions. The complexity increases in multi-tenant systems where many base models, users, applications, and tasks compete for the same resources. We anticipate future research will systematically advance solutions in this domain.

Service availability. As large generative models become essential infrastructure services, ensuring their resilience to failures is crucial. Unlike database services that incorporate logging, checkpointing, and replication in cloud-native architectures for both recoverability and availability, large model training on distributed GPU servers is resource-intensive and time-consuming [88]. Handling partial resource failures is vital to maintaining model performance [80]. Straw-man solutions like periodic checkpointing are costly and lead to GPU underutilization. Asynchronous approaches with fine-grained logging offer a more promising direction for fault-tolerant training, allowing for progress persistence without halting GPU execution. In contrast, online inference prioritizes availability over recoverability, as tasks are short and can be re-executed. Replicating inference deployments across multiple fault domains eliminates downtime but increases costs. Cloud-native proposals suggest cost optimizations through multi-tenancy, enabling replication for higher throughput and graceful failure recovery.

High availability is a challenge in LMaaS due to heterogeneous hardware, fragile machine learning software, and evolving user and application requirements. Future research opportunities include designing availability-guaranteed replication and recovery algorithms, hardware-software co-design for faster crash recovery, leveraging cloud storage architectures with specialized caching, and so on. Addressing these challenges is crucial as large language models become integral to various applications.

Resource accessability and ephemerality. Training and deploying large generative models often requires a multitude of GPUs, especially during foundational model pre-training. High-bandwidth networking is crucial for inter-node communication and gradient synchronization. However, this gives rise to resource accessibility issues, making it challenging to access such infrastructures. To address this, previous research has focused on training large models using geo-distributed infrastructures [86], and there are high expectations for further developments in this field. Simultaneously, cloud providers are also moving towards disaggregation. Global GPU market spaces like Vasi.ai and Akash Networks offer cost-effective rentals with highly heterogeneous compute nodes distributed worldwide. However, constructing cloud services on such hyper-disaggregated infrastructures poses significant challenges in communication, fault tolerance, and resource management. Nevertheless, the potential benefits are substantial.

Ephemeral cloud resources are increasingly prevalent, such as spot instances [7] where short-lived computational resources are provided and the rising zero-carbon clouds [21, 62] where the infrastructures rely heavily on sustainable energy sources (e.g., wind power). These resources often have factional prices compared to the rates for regular reserved or on-demand resources [8, 67] and can exhibit significant fluctuations over time. It is challenging to deploy LM tasks on ephemeral resources. Given that LM training is typically long-running, it is essential to establish an efficient mechanism for model checkpointing and reloading. This mechanism

allows training processes to be paused when resources become temporarily inaccessible or when prices become prohibitively high, and then resume at a more opportune time without bringing substantial overheads. LM inferences are often characterized by their sensitivity to latency; timely and reasonable decision-making becomes crucial for determining which inference jobs should be prioritized and given access to available resources.

Diverse microservices. In addition to the basic large model training, fine-tuning and inference services, recent production offerings start to explore broader microservices. For example, the lifecycle of machine learning contributes multiple useful microservices such as data curation, cleaning, transformation and labeling. Major cloud vendors and DBaaS such as Databricks [25] and Snowflake [22] already have relevant offerings and integration. A notable microservice for LMaaS that can be distinct to the aforementioned cases is the usage of vector databases [59] like Milvus [79] and Pinecone [1] to cache intermediate results, frequent queries, or embeddings [3, 51]. RAG-based applications, discussed earlier in this paper, is another important use case. Vector databases must choose the vector indexes (e.g., a graph-based index HNSW [50]) to tradeoff between memory consumption, query performance, and accuracy. Building robust, scalable and efficient vector database as microservices to the cloud system is of both research and production value.

AI operations (AIops) is now vital for managing complex cloud infrastructures. It automates tasks like monitoring, troubleshooting, performance tuning, and resource optimization. In cloud-based systems for large generative models, AIops will be a pivotal tool to ensure the system reliability and efficiency. Challenges and opportunities are both significant. For instance, predicting workload completion times and incoming workload patterns becomes crucial for enhancing system performance and minimizing service downtime [63, 78]. Optimizing the model training and inference configurations (e.g., batch sizes, optimizer parameters) based on the specific workload also has a significant impact on the system performance and resource consumption. Insights from database management systems, again, can be applied to intelligently automate the operation of LMaaS. For example, query forecasting methods [48] can help detect workload patterns and completion times. Learning-based knob-tuning techniques [6, 76] can also help optimize training and inference configurations. However, database and LM workloads have disparate properties, which warrants deeper investigation and adaptation of existing techniques.

Emerging applications and workloads. Apart from the RAG-based applications mentioned earlier in this paper, there have been emerging applications of large generative models; in addition to model inference calls, they often employ complex workflows and sometimes involve interactions or feedbacks from 3rd-party software such as compilers and search engines. One notable type of application is AI agents [47, 60] which require multi-turn interactions between the agent and human, or among multiple agents for web-shopping, robotics, gaming, and other use cases. Application builders create AI agents from large model inference endpoints [37, 73, 83] using different role-defining prompts. While many cloud-native techniques still apply in these scenarios, there are challenges and opportunities to optimize these novel workflows. For example, multiple AI agents that are fine-tuned with different skill sets from the same foundational model can be seen as another use case of multi-tenancy. Compute graph and query optimization techniques may be applied to improve the execution. We project that application-specific systems and optimizations will surge in the near future.

6 RELATED WORK

Systems for large models. PyTorch [61], TensorFlow [57], and JAX [16] are among the popular deep learning frameworks. PyTorch used a dynamic computation graph, known as the define-by-run paradigm, while TensorFlow was recognized for its static define-and-run approach; they now support both modes. JAX offers composable transformations of Python functions, facilitating advanced optimizations and just-in-time compilation to heterogeneous architectures. A layer above, systems that serve complex large models, especially in distributed setups [39, 53, 54], come to attention; Megatron-LM [56, 71], DeepSpeed [66, 72] and Huggingface Transformers [82] are three exemplars in this category that provide easy-to-use, distributed training and deployment support. To continuously serve large models with concurrent users and queries, Orca [85], TGI [74], and DeepSpeed-MII [26] build the model serving layer with gRPC user endpoints. These systems are necessary components of the AI-native computing paradigm.

Cloud-native in production. Major cloud providers offer a diverse range of cloud-native services. These include Database-as-a-Service (DBaaS) solutions like Amazon RDS [10], Azure SQL Databases [11]. Function-as-a-Service (FaaS) platforms like AWS Lambda [12], Azure Functions [13], and Google Cloud Functions [31] enable serverless code execution. Managed Kubernetes services such as Amazon EKS [9], Google Kubernetes Engine (GKE) [32], and Azure Kubernetes Service (AKS) [14] enable container orchestration. AI/ML services, big data and analytics, IoT platforms are among the offerings that cater to various business needs, demonstrating the versatility of cloud-native technologies.

7 CONCLUSIONS

This paper explores the similarities between Large-Model-as-a-Service (LMaaS) and Database-as-a-Service (DBaaS), uncovering shared characteristics and suggesting that cloud-native technologies can be leveraged to improve generative AI systems. However, creating an AI-native computing paradigm through this fusion demands deep insights and innovations. The paper presents three use cases with initial findings, emphasizing future research challenges and opportunities.

REFERENCES

[1] [n. d.]. Pinecone. ([n. d.]). https://www.pinecone.io/

[2] 2023. How Microsoft is Trying to Lessen Its Addiction to OpenAI as AI Costs Soar. https://www.theinformation.com/articles/how-microsoft-is-trying-to-lessen-its-addiction-to-openai-as-ai-costs-soar. (2023).

[3] 2023. LLM Limitations (https://zilliz.com/use-cases/llm-retrieval-augmented-generation). (2023).

[4] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Symposium on Operating Systems Design and Implementation (OSDI)*.

[5] Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. 2022. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems* 35 (2022), 22300–22312.

[6] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. {CherryPick}: Adaptively unearthing the best cloud configurations for big data analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 469–482.

[7] Amazon EC2 Spot Instances 2023. Amazon EC2 Spot Instances. https://aws.amazon.com/ec2/spot. (2023). Accessed: 2023-10-21.

[8] Amazon EC2 Spot Instances Pricing 2023. Amazon EC2 Spot Instances Pricing. https://aws.amazon.com/ec2/spot/pricing/. (2023). Accessed: 2023-10-21.

[9] Amazon EKS 2023. Amazon EKS. https://aws.amazon.com/eks/. (2023). Accessed: 2023-10-21.

[10] Amazon RDS 2023. Amazon RDS. https://aws.amazon.com/rds/. (2023). Accessed: 2023-10-21.

[11] Panagiotis Antonopoulos, Alex Budovski, Cristian Diaconu, Alejandro Hernandez Saenz, Jack Hu, Hanuma Kodavalla, Donald Kossmann, Sandeep Lingam, Umar Farooq Minhas, Naveen Prakash, et al. 2019. Socrates: The new sql server in the cloud. In *Proceedings of the 2019 International Conference on Management of Data*. 1743–1756.

[12] AWS Lambda 2023. AWS Lambda. https://aws.amazon.com/lambda/. (2023). Accessed: 2023-10-21.

[13] Azure Functions 2023. Azure Functions. https://azure.microsoft.com/en-us/products/functions. (2023). Accessed: 2023-10-21.

[14] Azure Kubernetes Service 2023. Azure Kubernetes Service. https://azure.microsoft.com/en-us/products/kubernetes-service. (2023). Accessed: 2023-10-21.

[15] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 141–159.

[16] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs. (2018). http://github.com/google/jax

[17] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering* 38, 4 (2015).

[18] Surajit Chaudhuri, Umeshwar Dayal, and Vivek Narasayya. 2011. An overview of business intelligence technology. *Commun. ACM* 54, 8 (2011), 88–98.

[19] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318* (2023).

[20] Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze Ceze, and Arvind Krishnamurthy. [n. d.]. Punica: Multi-Tenant LoRA Serving. *arXiv preprint arXiv:2310.18547*.

[21] Andrew A. Chien. 2021. Driving the Cloud to True Zero Carbon. *Commun. ACM* 64, 2 (2021), 5.

[22] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, et al. 2016. The snowflake elastic data warehouse. In *Proceedings of the 2016 International Conference on Management of Data*. 215–226.

[23] Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691* (2023).

[24] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.

[25] Databricks 2023. Databricks. https://www.databricks.com/. (2023). Accessed: 2023-10-21.

[26] DeepSpeed-MII 2023. DeepSpeed-MII. https://github.com/microsoft/DeepSpeed-MII. (2023). Accessed: 2023-10-21.

[27] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research* 23, 1 (2022), 5232–5270.

[28] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, et al. 2019. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 3–18.

[29] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. 2011. Dominant resource fairness: Fair allocation of multiple resource types. In *8th USENIX symposium on networked systems design and implementation (NSDI 11)*.

[30] Jonathan Goldstein and Per-Åke Larson. 2001. Optimizing queries using materialized views: a practical, scalable solution. *ACM SIGMOD Record* 30, 2 (2001), 331–342.

[31] Google Cloud Functions 2023. Google Cloud Functions. https://cloud.google.com/functions?hl=en. (2023). Accessed: 2023-10-21.

[32] Google Kubernetes Engine 2023. Google Kubernetes Engine. https://cloud.google.com/kubernetes-engine?hl=en. (2023). Accessed: 2023-10-21.

[33] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. 2020. Serving {DNNs} like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 443–462.

[34] Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, and Vidhya Srinivasan. 2015. Amazon redshift and the case for simpler data warehouses. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1917–1923.

[35] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. 2017. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409* (2017).

[36] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A platform for {Fine-Grained} resource sharing in the data center. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*.

[37] Sirui Hong, Xiawu Zheng, Jonathan P. Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zi Hen Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, and Chenglin Wu. 2023. MetaGPT: Meta Programming for Multi-Agent Collaborative Framework. *ArXiv* abs/2308.00352 (2023). https://api.semanticscholar.org/CorpusID:260351380

[38] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[39] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. 2019. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems* 32 (2019).

[40] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. 2017. Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the 2017 symposium on cloud computing*. 445–451.

[41] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. 2019. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383* (2019).

[42] Kubernetes 2023. Kubernetes. https://kubernetes.io/. (2023). Accessed: 2023-10-21.

[43] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180* (2023).

[44] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*. PMLR, 19274–19286.

[45] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[46] Beibin Li, Yao Lu, and Srikanth Kandula. 2022. Warper: Efficiently adapting learned cardinality estimators to data and workload drifts. In *Proceedings of the 2022 International Conference on Management of Data*. 1920–1933.

[47] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. AgentBench: Evaluating LLMs as Agents. *arXiv preprint arXiv: 2308.03688* (2023).

[48] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J Gordon. 2018. Query-based workload forecasting for self-driving database management systems. In *Proceedings of the 2018 International Conference on Management of Data*. 631–645.

[49] Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. 2020. Themis: Fair and efficient {GPU} cluster scheduling. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 289–304.

[50] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[51] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. 2023. High-Throughput Vector Similarity Search in Knowledge Graphs. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–25.

[52] Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. 2013. Naiad: A Timely Dataflow System. In *ACM Symposium on Operating Systems Principles (SOSP)*.

[53] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R Devanur, Gregory R Ganger, Phillip B Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized pipeline parallelism for DNN training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 1–15.

[54] Deepak Narayanan, Amar Phanishayee, Kaiyu Shi, Xie Chen, and Matei Zaharia. 2021. Memory-efficient pipeline-parallel dnn training. In *International Conference on Machine Learning*. PMLR, 7937–7947.

[55] Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. 2020. {Heterogeneity-Aware} cluster scheduling policies for deep learning workloads. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 481–498.

[56] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. 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*. 1–15.

[57] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. Tensorflow-serving: Flexible, high-performance ml serving. *arXiv preprint arXiv:1712.06139* (2017).

[58] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[59] James Jie Pan, Jianguo Wang, and Guoliang Li. 2023. Survey of Vector Database Management Systems. *CoRR* abs/2305.01087 (2023).

[60] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. (2023). arXiv:cs.HC/2304.03442

[61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019).

[62] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350* (2021).

[63] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, and Chuanxiong Guo. 2018. Optimus: an efficient dynamic resource scheduler for deep learning clusters. In *Proceedings of the Thirteenth EuroSys Conference*. 1–14.

[64] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems* 5 (2023).

[65] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International Conference on Machine Learning*. PMLR, 18332–18346.

[66] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3505–3506.

[67] Hossein Shafiei, Ahmad Khonsari, and Payam Mousavi. 2022. Serverless Computing: A Survey of Opportunities, Challenges, and Applications. *ACM Computing Survey* 54, 11s (2022), 239:1–239:32.

[68] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[69] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. 2019. Nexus: a GPU cluster engine for accelerating DNN-based video analysis. In *ACM Symposium on Operating Systems Principles (SOSP)*.

[70] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. (2023).

[71] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[72] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990* (2022).

[73] Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. (2023). arXiv:cs.AI/2306.03314

[74] TGI 2023. TGI. https://github.com/huggingface/text-generation-inference. (2023). Accessed: 2023-10-21.

[75] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[76] Dana Van Aken, Andrew Pavlo, Geoffrey J Gordon, and Bohan Zhang. 2017. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM international conference on management of data*. 1009–1024.

[77] Shivaram Venkataraman, Aurojit Panda, Kay Ousterhout, Michael Armbrust, Ali Ghodsi, Michael J Franklin, Benjamin Recht, and Ion Stoica. 2017. Drizzle: Fast and adaptable stream processing at scale. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 374–389.

[78] Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. 2016. Ernest: Efficient performance prediction for {Large-Scale} advanced analytics. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 363–378.

[79] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. 2614–2627.

[80] Zhuang Wang, Zhen Jia, Shuai Zhang, Zhen Zhang, Mason Fu, TS Eugene Ng, and Yida Wang. 2023. Gemini: Fast failure recovery in distributed training with in-memory checkpoints. (2023).

[81] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[82] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[83] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. arXiv:cs.AI/2308.08155

[84] Yongji Wu, Matthew Lentz, Danyang Zhuo, and Yao Lu. 2022. Serving and Optimizing Machine Learning Workflows on Heterogeneous Infrastructures. *arXiv preprint arXiv:2205.04713* (2022).

[85] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. 521–538.

[86] Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. 2022. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems* 35 (2022), 25464–25477.

[87] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. 2013. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the twenty-fourth ACM symposium on operating systems principles*. 423–438.

[88] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).