

Cloud-Native Systems for Generative AI Applications: Current Trends and Open Challenges

Minchen Yu¹, Wei Wang², Yue Cheng³, Hong Xu⁴

¹The Chinese University of Hong Kong, Shenzhen

²Hong Kong University of Science and Technology

³University of Virginia

⁴The Chinese University of Hong Kong

1 ABSTRACT

In recent years, Generative Artificial Intelligence (GAI) has advanced rapidly and seen widespread adoption across a broad range of domains, placing unprecedented demands on cloud systems and infrastructure. We identify three key trends—spanning the model, application, and infrastructure levels—that are collectively driving a shift toward *modular design* for GAI systems, akin to the microservice architecture widely used in cloud-native platforms. In this paradigm, GAI workloads are decomposed into specialized, fine-grained modules that can be deployed, managed, and scaled independently. This microservice-like modular approach enables precise resource allocation, efficient sharing of heterogeneous hardware, and improved scalability in response to dynamic workload characteristics and evolving application requirements. Building on this design principle, we outline several open challenges that warrant further investigation in future research.

2 MODULAR DESIGN FOR GAI APPLICATIONS

The rapid evolution of GAI inference systems positions *modular design* as a unifying architectural principle for next-generation, cloud-native GAI applications.

(1) Model-level: activation sparsity and multiple modality. Large models continue to grow in parameter scale and capability, driving the need for new algorithms and execution strategies. (i) Activation sparsity (e.g., Mixture-of-Experts (MoE) [4] and sparse attention mechanisms [10]) selectively activates only the most relevant components for a given input. (ii) Multi-modal models (e.g., GPT-5 [1] and Gemini [3]) employ distinct encoders for different inputs (text, vision, audio, structured data), each with unique resource needs. This intrinsic heterogeneity naturally motivates disaggregation of inference into stage-specific or component-specific deployments, characterized by several examples. (i) *Prefill/Decoding (P/D) disaggregation* separates the compute-bound prefill from the memory-bound decoding, allowing resource allocation to be tailored for each stage [8, 12]. (ii) *Attention/FFN (A/F) disaggregation* decouples attention layers that require maintaining KV cache states from stateless FFN/MoE layers, enabling independent resource allocation and scaling [9, 13]. (iii) *modality-specific deployment* enables vision, audio, or structured-data modules to run independently, with optimal placement and scheduling for their workloads [11].

(2) Application-level: multi-agent systems. At the application level, multi-agent GAI uses large models as autonomous agents—capable of reasoning, planning, tool use, and coordination—to execute complex, multi-stage workflows [2, 5]. These systems are service-oriented, with specialized roles (planner, executor, verifier), retrieval components (e.g., RAG), and orchestration layers. The

functional decomposition maps naturally to a modular architecture, where each component runs as an independent microservice with its own lifecycle and scaling. Such modularization provides several key benefits. (i) Failures in one agent or service are contained, preventing cascading errors. (ii) Individual components can scale out independently. (iii) Fine-grained control over component placement enhances data locality and resource sharing across various hardware accelerators.

(3) Infrastructure-level: resource heterogeneity and high-speed interconnects. Rising AI workload demands have driven rapid evolution in hardware accelerators, resulting in greater heterogeneity within cloud environments [7]. This spans multiple GPU generations from the same vendor (e.g., Nvidia H100 vs. A100, differing in compute, memory, and interconnect topology) and fundamentally different architectures from AMD GPUs, Google TPUs, and AI-specific ASICs. Concurrently, modern AI data centers now feature high-speed, low-latency interconnects that enable: (i) rack-scale systems (e.g., Nvidia GB200 NVL72 SuperPOD, Huawei CloudMatrix) linking hundreds or thousands of accelerators as a unified compute resource [14]; (ii) dynamic resource pooling via NVLink, NVSwitch, InfiniBand, and CXL, allowing flexible sharing of compute, memory, and heterogeneous accelerators [6]. These infrastructure trends create significant opportunities for modular GAI system design. Disaggregated modules and services can be deployed on heterogeneous accelerators that best align with their computational and memory characteristics; high-speed interconnects allow these fine-grained modules to efficiently operate on separated hardware without prohibitive data transfer overhead; and dynamic resource pooling supports on-demand cluster reconfiguration for precise scaling with minimal waste.

3 CLOUD-NATIVE SYSTEMS FOR GAI

3.1 Goal

Building on the modular design discussed above, a critical question arises: *Can we design cloud-native platforms that streamline the development, deployment, and research of generative AI applications—serving a role similar to that of Kubernetes for microservices?* As illustrated in Fig. 1, such a platform should natively support modular, disaggregation-friendly AI applications, allowing flexible, fine-grained model deployment. It must not only ease execution of current workloads but also unlock new applications that exploit modularity, while ensuring efficient use of heterogeneous accelerators and high-speed interconnects. We highlight three main design goals. (1) *Usability*: Enabling developers and researchers to easily build and deploy modular, disaggregated solutions without

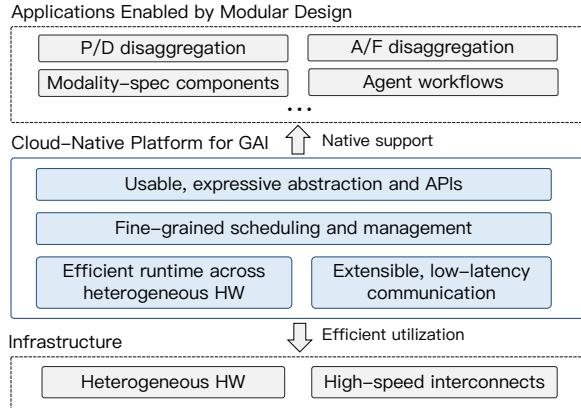


Figure 1: Overview of the cloud-native platform for GAI.

managing low-level infrastructure. (2) *Scalability*: Scaling up and out efficiently to meet changing workload demands by exploiting high-speed interconnects and intelligent scheduling for consistent, high performance. (3) *Resource efficiency*: Supporting fine-grained task packing, automated placement, and resource sharing in a multi-tenant environment to maximize utilization and reduce costs.

3.2 Open Challenges

Recent AI cloud platforms—such as AWS Bedrock, Google Vertex AI, Lambda Cloud, RunPod, Together AI, and llm-d—offer specialized infrastructure for large-scale GAI deployment. They partially bridge the gap between general-purpose cloud services and the stringent performance, scalability, and flexibility needs of modern GAI. However, significant technical challenges remain.

(1) Usable, expressive abstraction and APIs. Current platforms generally expose functionality at two abstraction levels. (i) *Resource-level abstraction* enables direct allocation of GPU instances or containers, offering flexibility for custom software stacks but requiring deep expertise in deployment and optimization. (ii) *Application-level abstraction* provides APIs for hosted inference APIs (e.g., REST endpoints for a finetuned LLM) that hide hardware and infrastructure details but limit control over model partitioning, scaling, and hardware mapping. These extremes make it difficult to support modular, disaggregated inference—such as placing prefill and decoding on separate clusters or isolating modality-specific encoders. Ideally, the cloud-native planform should be *high-level yet fine-grained*, with: (i) *New abstraction layer*: The platform should provide a composable deployment model at the granularity of model components or pipeline stages, allowing high-level control without direct resource management. (ii) *Performance-oriented APIs*: The interfaces should allow developers to specify throughput, latency, or cost targets instead of explicitly selecting from a list of hardware types, enabling the platform to automatically map these requirements to suitable heterogeneous resources. (iii) *Rich dataflow specifications*: The system should natively support the expression and management of complex, dynamic dependencies, such as KV-cache streaming between P/D instances or cross-modal feature fusion.

(2) Fine-grained scheduling and management. Modular, disaggregated deployments bring greater flexibility but also introduce

new challenges for scheduling, orchestration, and resource management. Traditional coarse-grained strategies—scaling entire applications or models—are ill-suited when different modules have distinct performance characteristics and complex inter-dependencies. We identify three key requirements. (i) *Logical decomposition, physical composition*: The platform should allow logical separation of modules for design flexibility, while enabling the scheduler to co-locate or merge services on heterogeneous hardware that best matches their resource profiles; this approach improves overall utilization while sustaining end-to-end performance. (ii) *Fine-grained scaling*: It should dynamically scale individual bottleneck modules independently (e.g., P/D instances, modal-specific generators), avoiding the cost of over-provisioning the entire application. (iii) *Fault handling*: In a modular architecture, failures can quickly propagate due to the fine-grained interactions among components. Therefore, the system should detect faults promptly and contain them within the affected module, preventing their spread and minimizing disruption to other unaffected components of the application.

(3) Efficient runtime across heterogeneous hardware. AI cloud infrastructure increasingly integrates diverse accelerators across vendors and generations (e.g., GPUs and NPUs). The central challenge is to provide a runtime layer that can exploit this diversity without imposing excessive complexity or performance penalties. The runtime should deliver: (i) *Transparency*: The runtime should enable seamless execution across a variety of accelerators, allowing the same module or pipeline to run without modification on different devices. (ii) *Low overhead*: The runtime should minimize startup latency, support rapid scaling up or down, and facilitate efficient migration of modules or tasks between accelerators. (iii) *Flexible isolation*: The runtime should provide multiple levels of isolation to suit different scenarios. It should enforce strong isolation across jobs in multi-tenant environments for performance and security, while allowing more relaxed boundaries between modules within a single application to enable resource sharing, improve data locality, and accelerate inter-module communication.

(4) Extensible, low-Latency communication. Fine-grained, modular deployment significantly increases the volume, dynamism, and diversity of inter-module communication patterns. Existing communication frameworks—such as NCCL, NVSHMEM, or NIXL—are typically optimized for fixed, well-structured traffic patterns (e.g., a predefined set of collective primitives or point-to-point communications). However, these solutions do not efficiently handle dynamic topologies or irregular, evolving traffic, both of which are common in modern inference pipelines where modules may be scaled, replicated, or migrated at runtime. We identify two critical gaps. (i) *Programmable, extensible communication layer*: The platform should expose an interface that allows developers to define new message types, routing policies, and in-flight data processing strategies (e.g., compression, transformation). This flexibility enables the system to adapt communication behavior to diverse cross-module interaction patterns and shifting workload dynamics. (ii) *High performance*: The communication layer should automatically optimize transfers over different hardware and network fabrics, including NVLink, InfiniBand, PCIe, and Ethernet. This optimization should encompass dynamic path selection, adaptive load balancing, and congestion-aware routing, ensuring consistent performance even as modules are placed on varying interconnects.

4 CONCLUSION

We discuss cloud-native systems for GAI that embrace modular design principles. (1) For developers and researchers, this approach simplifies deployment of large GAI applications, enabling an optimal balance between performance and cost across heterogeneous resources. (2) For cloud providers, it enables fine-grained task scheduling and resource sharing, improving overall utilization. We aim for this discussion to inspire further exploration and innovation towards usable, scalable, and efficient GAI cloud platforms.

ACKNOWLEDGEMENT

We thank Yizhou Shan for his helpful comments.

REFERENCES

- [1] [n. d.]. OpenAI’s GPT-5. <https://openai.com/index/introducing-gpt-5/>.
- [2] Lingjiao Chen, Jared Davis, Boris Hanin, Peter Balis, Ion Stoica, Matei Zaharia, and James Zou. 2025. Are more LM calls all you need? towards the scaling properties of compound AI systems. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.
- [3] Gheorghe Comanici et al. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *arXiv preprint arXiv:2507.06261* (2025).
- [4] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. *arXiv preprint arXiv:2401.06066* (2024).
- [5] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. *arXiv preprint arXiv:2402.01680* (2024).
- [6] Heng Liao, Bingyang Liu, Xianping Chen, Zhigang Guo, Chuanning Cheng, Jianbing Wang, Xiangyu Chen, Peng Dong, Rui Meng, Wenjie Liu, Zhe Zhou, Ziyang Zhang, Yuhang Gai, Cunle Qian, Yi Xiong, Zhongwu Cheng, Jing Xia, Yuli Ma, Xi Chen, Wenhua Du, Shizhong Xiao, Chungang Li, Yong Qin, Liudong Xiong, Zhou Yu, Lv Chen, Lei Chen, Buyun Wang, Pei Wu, Junen Gao, Xiaochu Li, Jian He, Shizhuhan Yan, and Bill McColl. 2025. UB-Mesh: a Hierarchically Localized ND-FullMesh Datacenter Network Architecture. *arXiv preprint arXiv:2503.20377* (2025).
- [7] Yixuan Mei, Yonghao Zhuang, Xupeng Miao, Juncheng Yang, Zhihao Jia, and Rashmi Vinayak. 2025. Helix: Serving Large Language Models over Heterogeneous GPUs and Network via Max-Flow. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 586–602.
- [8] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture*. 118–132.
- [9] Ao Xiao et al. 2025. xDeepServe: Model-as-a-Service on Huawei CloudMatrix384. *arXiv preprint arXiv:2508.02520* (2025).
- [10] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention. *arXiv preprint arXiv:2502.11089* (2025).
- [11] Zili Zhang, Yimin Zhong, Ranchen Ming, Hanpeng Hu, Jianjian Sun, Zheng Ge, Yibo Zhu, and Xin Jin. 2024. DistTrain: Addressing Model and Data Heterogeneity with Disaggregated Training for Multimodal Large Language Models. *arXiv preprint arXiv:2408.04275* (2024).
- [12] Yimin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: disaggregating prefill and decoding for goodput-optimized large language model serving. In *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*.
- [13] Ruidong Zhu, Ziheng Jiang, Chao Jin, Peng Wu, Cesar A. Stuardo, Dongyang Wang, Xinlei Zhang, Huaping Zhou, Haoran Wei, Yang Cheng, Jianzhe Xiao, Xinyi Zhang, Lingjun Liu, Haibin Lin, Li-Wen Chang, Jianxi Ye, Xiao Yu, Xuanzhe Liu, Xin Jin, and Xin Liu. 2025. MegaScale-Infer: Serving Mixture-of-Experts at Scale with Disaggregated Expert Parallelism. *arXiv preprint arXiv:2504.02263* (2025).
- [14] Pengfei Zuo et al. 2025. Serving Large Language Models on Huawei CloudMatrix384. *arXiv preprint arXiv:2506.12708* (2025).