Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

General-Purpose Cloud Compute For Agentic AI: Architectural Evolution For Scalable Inference, Reasoning, and Orchestration

Priyadarshni Shanmugavadivelu

Birla Institute of Technology and Science, Pilani, India.

ABSTRACT
The rapid evolution of generative artificial intelligence is transforming the requirements placed on cloud infrastructure. While graphics processing units (GPUs) remain central to accelerating large language model (LLM) inference, the emergence of Agentic AI systems introduces demands that extend well beyond accelerator throughput. Agentic AI performs iterative reasoning, long-horizon planning, memory retrieval, tool invocation, and multi-agent collaboration, requiring continuous coordination across compute, networking, storage, and virtualization layers that traditional inference-only workloads never exercised. This paper examines how general-purpose cloud compute has architecturally evolved to support these coordination demands. It synthesizes recent literature on agentic reasoning frameworks, LLM inference-serving systems, memory-augmented agent architectures, and composable infrastructure technologies such as Compute Express Link (CXL) to develop the Agentic AI Compute Stack, a five-layer conceptual framework describing how general-purpose compute operates as the orchestration backbone connecting enterprise applications to AI accelerators. The paper further analyzes workload-aware scheduling, heterogeneous CPU-GPU coordination, and memory-management techniques as the specific architectural mechanisms that determine whether agentic workloads scale efficiently. It argues that the future scalability of Agentic AI depends on optimizing the complete infrastructure stack rather than accelerator performance in isolation, and that this reframing carries direct implications for how cloud providers architect, schedule, and provision the infrastructure underneath autonomous AI systems. The discussion concludes by identifying open architectural questions for infrastructure research as agentic workloads continue to grow in scale and autonomy.

KEYWORDS Agentic AI, Cloud Computing, General-Purpose Compute, LLM Inference Serving, Virtualization, Workload Orchestration, Compute Express Link.

1. INTRODUCTION

Generative AI has progressed rapidly from single-turn, inference-only applications toward autonomous systems capable of planning, reasoning, and executing multi-step tasks with limited human intervention [1][2][3]. Much of this progress traces back to the finding that decomposing a problem into intermediate reasoning steps, rather than requiring a model to produce a final answer directly, substantially improves a language model's ability to perform complex, multi-step reasoning, a technique now commonly referred to as chain-of-thought prompting [4]. These systems, broadly termed Agentic AI, combine large language models with memory components, external tool access, retrieval mechanisms, and decision-making control loops that allow a single user request to expand into a chain of dependent computational steps [3][5]. Frameworks such as ReAct demonstrated that interleaving reasoning traces with concrete actions allows a language model to interact with external environments and revise its own plans mid-execution, building directly on the chain-of-thought insight that intermediate reasoning steps improve task performance [1][4], while Reflexion showed that agents can improve iteratively by verbally critiquing their own prior attempts and storing that critique for use in later trials [5]. Toolformer extended this trajectory by showing that a language model can learn, in a largely self-supervised manner, when and how to invoke external tools such as calculators, search engines, and APIs [2]. Multi-agent frameworks such as AutoGen subsequently generalized this pattern by allowing multiple specialized agents to converse with one another, delegate subtasks, and collectively resolve problems that a single agent could not resolve alone [6].

The practical consequence of this progression is architectural rather than purely algorithmic. Unlike a conventional inference request, which begins and ends with a single forward pass through a model, an agentic request may trigger retrieval against an enterprise knowledge base, one or more tool calls, several rounds of internal reasoning, a memory read-write cycle, and coordination with other agents before a final response is produced [7][8]. Each of these steps consumes general-purpose compute resources for scheduling, orchestration, networking, storage access, and security enforcement, even though the neural network computation itself is offloaded to GPUs or other accelerators [9]. As agentic systems are deployed at production scale, this orchestration overhead becomes a first-order factor in overall system latency, reliability, and cost, rather than a secondary implementation detail [8].

This paper argues that sustaining Agentic AI at scale requires addressing architectural coordination bottlenecks alongside accelerator-performance constraints, rather than focusing on accelerator performance in isolation. Prior discussions of AI infrastructure have tended to concentrate on GPU throughput, memory bandwidth, and interconnect speed within the accelerator itself [10][11]. That focus, while necessary, is incomplete: recent systems research on LLM inference serving demonstrates that memory management, request scheduling, and CPU-GPU coordination are themselves major determinants of achievable throughput, independent of any single accelerator's raw compute capability [12][13][14][15]. Agentic workloads amplify this dependency further, because they introduce variable, unpredictable, and often long-running execution patterns that traditional batch-oriented cloud scheduling was not designed to handle [8][16].

This distinction matters because the two categories of bottleneck respond to different interventions. An accelerator-bound bottleneck is resolved by faster or more numerous GPUs; a coordination-bound bottleneck is resolved by better scheduling, memory management, or orchestration logic, and no amount of additional accelerator capacity will resolve it. Treating Agentic AI infrastructure planning as though every bottleneck were accelerator-bound risks systematic overinvestment in the wrong layer of the stack, while the coordination bottlenecks that actually constrain end-to-end latency and reliability remain unaddressed.

Existing research has examined agent frameworks, inference-serving systems, memory architectures, and composable infrastructure largely as separate technical domains. However, the literature lacks a unified architectural model that explains how these components interact to support end-to-end agentic execution and clarifies the specific coordination role played by general-purpose cloud compute. First, it synthesizes recent literature spanning agentic reasoning frameworks, LLM-serving infrastructure, agent memory systems, and composable hardware technologies into a single architectural narrative, rather than treating these as separate research threads, several of which have so far developed largely independently of one another despite addressing complementary parts of the same underlying infrastructure problem. Second, it introduces the Agentic AI Compute Stack, a five-layer conceptual framework that unifies these previously fragmented research threads and makes explicit how general-purpose compute coordinates enterprise applications, AI orchestration frameworks, accelerators, and physical infrastructure.

This paper adopts a structured narrative review approach appropriate to a conceptual, literature-synthesis contribution rather than a systematic review. Sources were drawn primarily from ACM and USENIX systems venues (SOSP, OSDI, NSDI, ATC), NeurIPS and ICLR proceedings, IEEE and arXiv preprints, and foundational surveys on cloud computing and hardware acceleration, with publication dates concentrated between 2020 and 2025 to capture the period in which agentic and LLM-serving research matured. Search themes included agentic reasoning frameworks, LLM inference serving, KV-cache and memory management, workload-aware scheduling, and composable infrastructure. Representative studies were selected for methodological influence, citation prominence, and direct relevance to the coordination problem this paper addresses, rather than for exhaustive topical coverage.

The remainder of the paper is organized as follows. Section 2 reviews the evolution of general-purpose cloud compute in response to Agentic AI. Section 3 discusses cloud resource optimization techniques virtualization, workload-aware scheduling, and memory technologies that support agentic workloads. Section 4 presents the Agentic AI Compute Stack. Section 5 discusses future directions, and Section 6 concludes.

2. EVOLUTION OF GENERAL-PURPOSE CLOUD COMPUTE FOR AGENTIC AI

Cloud computing has evolved through several distinct phases in its relationship to artificial intelligence workloads. Early cloud platforms were built primarily to virtualize and elastically allocate general-purpose compute for enterprise applications [17][18][19]. Virtualization, in this earlier era, was concerned mainly with multi-tenant isolation and hardware utilization: hypervisors allowed multiple workloads to share physical servers safely, and elastic provisioning allowed capacity to track demand without manual intervention

[17][19]. The foundational cloud-computing literature framed this shift primarily as an economic and operational one, converting fixed capital investment in hardware into elastic, pay-as-you-go operational expenditure, with the underlying infrastructure treated largely as a commoditized, interchangeable resource pool [18]. As cloud providers diversified their offerings, virtual machine families became increasingly specialized, with some optimized for compute-intensive workloads and others for memory capacity, storage throughput, or network bandwidth, reflecting a broader recognition that a single general-purpose configuration could not efficiently serve all workload types [18][16]. This specialization already represented an early departure from the commodity-resource-pool assumption, since it required cloud providers to reason explicitly about which workload characteristics warranted which hardware configuration, a precursor to the far more granular workload-awareness that Agentic AI now demands.

The emergence of deep learning shifted this trajectory. Early AI infrastructure was overwhelmingly oriented toward offline model training, where the dominant concern was raw computational throughput over long-running batch jobs [10][20]. As trained models moved into production, however, the dominant workload shifted to online inference, which introduced strict latency, availability, and scalability requirements that batch-oriented infrastructure was not originally designed to meet [12][21]. Systems research responded with inference-specific serving architectures: vLLM's PagedAttention technique, for example, addressed the specific problem of wasted GPU memory in key-value (KV) cache management by borrowing virtual-memory paging concepts from operating systems, achieving substantial throughput improvements over earlier serving frameworks without any change to the underlying model [12]. Around the same period, systems such as S3 addressed GPU underutilization caused by conservative worst-case memory reservation, using output-length prediction to pack more concurrent requests onto the same hardware [13]. What unites both contributions is that neither altered the underlying model architecture at all; both achieved substantial throughput gains purely through better coordination of existing accelerator memory and scheduling.

Agentic AI represents a further escalation of these infrastructure demands. Rather than a single inference call, an agentic workload typically consists of a variable-length chain of model calls, tool invocations, and memory operations, each of which may depend on the output of the previous step [3][7]. This chaining behavior means that the traditional metrics used to provision inference infrastructure, namely fixed request latency and throughput under steady load, no longer fully capture the workload's resource profile. A single user-facing agentic request may internally resolve into a planning call, one or more retrieval operations, several tool invocations each with its own latency and failure profile, and a final synthesis call, meaning that the effective load an agentic application places on the underlying infrastructure can be substantially larger and more variable than a simple request-count metric would suggest. Recent serving systems have begun to address this directly. Fiddler, for instance, targets the specific inefficiency created by sparse mixture-of-experts (MoE) model architectures, using CPU compute to reduce the volume of data that must move between CPU and GPU memory during inference, achieving an order-of-magnitude improvement over prior approaches for uncompressed MoE models on constrained GPU memory [14]. Similarly, dLoRA addresses the serving inefficiency created by systems that must support many different low-rank adaptation (LoRA) variants of a base model simultaneously, a pattern increasingly common in multi-agent deployments where different agents may be specialized via lightweight fine-tuning, by dynamically merging and migrating adapters across worker replicas rather than statically assigning them [15]. Both systems illustrate a broader pattern relevant to Agentic AI specifically: as the number of distinct model variants, tool integrations, and concurrent agent sessions grows, the coordination problem of deciding which requests share which resources, and when, becomes at least as consequential as the raw compute available to service any single request.

Processor architecture has evolved in parallel with these serving-layer innovations. Modern server-class CPUs increasingly emphasize higher core counts, deeper cache hierarchies, greater memory bandwidth, and faster interconnects, including PCIe Gen5 and Compute Express Link (CXL), rather than pursuing single-core clock speed as the primary axis of improvement [9][10]. CXL in particular is significant for agentic workloads because it enables memory pooling and expansion across multiple servers, addressing a specific bottleneck that grows more severe as agentic sessions extend in length: every additional token retained in a model's context window requires key-value cache memory, and a single long-running agentic session, particularly one that fans out into multiple chained sub-calls, each carrying its own context, can consume memory on a scale that a single server's local memory was never provisioned to hold [10]. Because CXL operates at the physical-interconnect layer rather than within any individual accelerator, it is a clear example of an infrastructure innovation that improves agentic-workload scalability without touching accelerator design at all. The result of these combined trends is a heterogeneous computing environment in which CPUs increasingly specialize in orchestration, coordination,

and memory management, while GPUs and other accelerators specialize in the parallel numerical computation intrinsic to neural network execution [10][9]. This division of labor is not merely a matter of hardware specialization; it reflects a genuine difference in the computational character of the two categories of work. Neural network inference is highly parallel and largely stateless from one token to the next within a single forward pass, which is precisely what accelerator architectures are optimized to exploit. Orchestration logic, by contrast, is inherently sequential and stateful: a scheduler must know the outcome of one step before deciding the next, which is exactly the kind of workload general-purpose processors, with their emphasis on branch prediction, cache hierarchies, and low-latency single-thread execution, are built to handle efficiently. Recognizing this division of labor as a stable architectural principle, rather than a temporary artifact of current hardware limitations, is part of why this paper treats general-purpose compute as a durable, first-order concern for Agentic AI infrastructure rather than a layer likely to be subsumed by future accelerator generations.

3. CLOUD RESOURCE OPTIMIZATION FOR AGENTIC AI

Efficient resource management has become a defining capability of infrastructure intended to support Agentic AI, precisely because agentic workloads exhibit execution patterns that are far more variable than either traditional enterprise applications or single-turn inference requests [8][16]. Cloud platforms must therefore allocate compute resources dynamically while continuing to meet latency and cost constraints that do not relax simply because a workload has become more complex.

Virtualization remains foundational to this process, but its role has shifted. Where early cloud virtualization was concerned primarily with static multi-tenant isolation, contemporary schedulers increasingly incorporate live workload characteristics, processor topology, and memory locality directly into placement decisions [17][16]. Non-uniform memory access (NUMA)-aware scheduling has become particularly important as core counts per processor continue to grow: maintaining locality between the CPU cores executing an agent's orchestration logic and the memory holding its working state reduces latency for exactly the kind of control-plane operations, such as planning, tool-call dispatch, and memory lookups, that agentic workloads perform far more frequently than conventional applications [9][16]. The sensitivity of agentic workloads to this kind of locality is easy to underestimate, because any individual orchestration operation is computationally trivial compared to a single forward pass through a large model; the risk is not that any one operation is slow in isolation, but that an agentic request may involve dozens of such operations chained together, so that even small, per-operation memory-locality penalties compound into meaningful end-to-end latency across the full execution chain. This compounding effect is precisely what distinguishes agentic workloads from conventional inference workloads with respect to topology-aware scheduling. A single-turn inference request typically executes as one bounded, largely self-contained computation, so a locality penalty on any individual operation has limited opportunity to propagate; a long-running, multi-agent execution graph, by contrast, chains together planning steps, tool calls, memory reads and writes, and inter-agent messages that may span many CPU cores and memory domains over an extended session, giving topology mismatches many more opportunities to accumulate. Topology-aware scheduling is therefore not merely a marginal latency optimization for agentic infrastructure, as it might be for a conventional workload, but a structural requirement for keeping end-to-end latency bounded as session length and agent concurrency grow.

Machine-learning-centric approaches to resource management have also matured substantially. Rather than treating scheduling as a static rule-based problem, recent surveys document a shift toward learned scheduling policies that incorporate workload concurrency, request characteristics, and predicted resource needs directly into placement and autoscaling decisions [16][20]. This shift is well suited to agentic workloads specifically because the branching, unpredictable structure of an agent's execution graph is difficult to characterize with the fixed heuristics that sufficed for steadier workloads [20]. A conventional web-service scheduler, for instance, can often assume that requests are roughly independent and similarly sized; an agentic scheduler cannot make either assumption, since a single top-level request may fan out into a dependency graph of sub-tasks whose size and duration are not known until earlier steps in the chain have already completed. AI-driven job scheduling research has similarly moved toward predictive and adaptive approaches that account for the heterogeneity of modern cloud resource pools, rather than relying solely on CPU utilization as a scheduling signal [20][21]. These predictive approaches are directly relevant to agentic infrastructure because they treat workload behavior as something to be forecast and planned around rather than reacted to after the fact, which is a closer match to the bursty, dependency-driven demand pattern that multi-step agentic execution produces.

Memory-management innovations complement these scheduling advances. As discussed in Section 2, CXL-enabled memory pooling directly addresses the growing memory footprint of long-running or multi-agent sessions by allowing memory to be expanded and shared across servers rather than confined to a single machine's physical capacity [10]. This capability is likely to become increasingly consequential as agentic applications support larger working-memory footprints and greater numbers of concurrent autonomous agents operating over shared enterprise context [9][10]. Without such pooling, a cloud operator supporting many concurrent agentic sessions faces an uncomfortable choice between over-provisioning every server for worst-case memory demand, which is costly and often idle capacity in practice, or accepting hard ceilings on session length and concurrency that directly constrain the sophistication of the agentic applications that can be deployed.

Agent memory architectures represent a related but distinct optimization concern, situated at the application layer rather than the infrastructure layer, yet with direct infrastructure implications. Retrieval-augmented generation (RAG) architectures, for example, allow a model to incorporate external knowledge at inference time rather than relying solely on parameters learned during training, improving factual accuracy for knowledge-intensive tasks while introducing additional retrieval and indexing infrastructure requirements [7]. Because retrieval typically occurs on a separate path from the model's own forward pass, RAG architectures place additional demand on storage indexing, network bandwidth between the retrieval store and the inference service, and the general-purpose compute responsible for coordinating the two, demand that scales with the frequency and complexity of retrieval calls rather than with model size. More recent agentic memory systems go further, proposing that an agent's memory should be organized dynamically rather than stored as a flat log, using techniques such as automatically generated contextual notes and dynamic linking between related memories to construct an evolving, interconnected knowledge structure that the agent can draw on across sessions [22]. Systems of this kind trade a larger and more complex memory-management workload, borne by general-purpose compute, for improved reasoning continuity at the model layer, a trade-off that is only favorable if the underlying infrastructure can absorb the added indexing and retrieval overhead without degrading response latency. Each of these memory mechanisms adds a further layer of storage, indexing, and retrieval demand onto the general-purpose compute layer.

Taken together, these optimization techniques, namely topology-aware virtualization and scheduling, learned and predictive resource management, and composable memory infrastructure, illustrate that resource optimization for Agentic AI is a multi-layered, cross-cutting concern rather than a single point of improvement. Progress in any one layer in isolation, without corresponding coordination across the others, is unlikely to yield proportional gains in overall system efficiency [18][21]. A scheduler that places workloads optimally but operates on a server whose memory cannot expand to hold a growing agentic session's context will still bottleneck; conversely, ample pooled memory cannot compensate for a scheduler that repeatedly places latency-sensitive orchestration logic far from the memory it depends on. It is this interdependency across layers, more than any single technique in isolation, that the Agentic AI Compute Stack introduced in the next section is intended to make explicit. The optimization techniques reviewed above are drawn from literatures that have so far developed largely in isolation from one another. Virtualization and scheduling research, systems work on inference serving, and agent-memory research each address a piece of the infrastructure problem, but none individually models how applications, orchestration, general-purpose compute, accelerators, and physical infrastructure interact across a complete agentic execution. A unified framework is needed precisely because these existing models describe individual parts of the stack rather than the end-to-end coordination that determines whether an agentic workload actually scales.

4. THE AGENTIC AI COMPUTE STACK

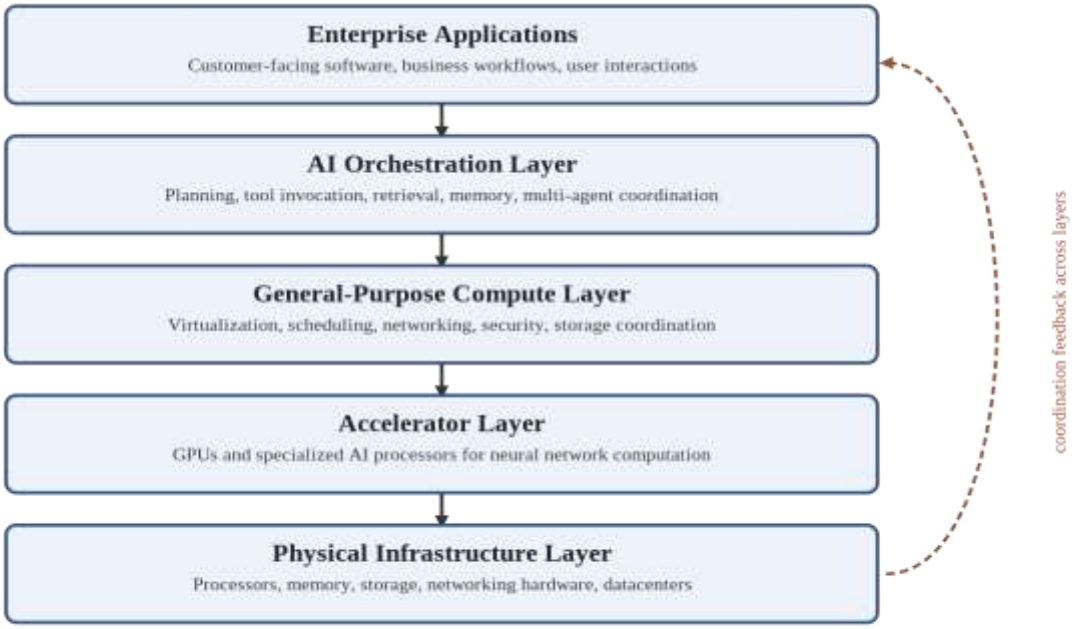

Figure 1. The Agentic AI Compute Stack: five cooperating infrastructure layers.

Figure 1. The Agentic AI Compute Stack. (Author's original synthesis based on the literature reviewed in Sections 2 and 3.)

The preceding sections establish that Agentic AI workloads place demands on cloud infrastructure that extend well beyond accelerator performance, spanning virtualization, scheduling, memory management, and inter-agent coordination. This section introduces the Agentic AI Compute Stack, a conceptual framework synthesizing these demands into a single layered model of how general-purpose compute and AI accelerators cooperate to execute autonomous AI workloads.

The novelty of this framework lies in its integration rather than in any single component: existing literature treats cloud infrastructure, inference-serving systems, and agent frameworks largely as independent domains, each analyzed with its own metrics and its own notion of what counts as a bottleneck. The Agentic AI Compute Stack instead integrates these domains into a single architectural model centered on cross-layer coordination, making explicit the dependencies between layers that a domain-siloed treatment leaves implicit. The framework consists of five logical layers. The Enterprise Applications layer represents the customer-facing software, business workflows, and user interactions that initiate agentic requests. The AI Orchestration layer manages planning, workflow execution, tool invocation, retrieval, memory read-write operations, and multi-agent coordination; it is the layer most directly responsible for implementing the reasoning-and-acting patterns established in frameworks such as ReAct and AutoGen [1][6]. The General-Purpose Compute layer provides virtualization, workload-aware scheduling, networking, security enforcement, storage coordination, and resource management; this layer functions as the platform's operational backbone, and is the layer whose evolution this paper has principally examined [17][16]. The Accelerator layer contains the GPUs and specialized AI processors responsible for the computationally intensive neural network operations underlying inference, supported by serving-layer innovations such as PagedAttention and Fiddler that determine how efficiently accelerator capacity is actually used [12][14]. Finally, the Physical Infrastructure layer encompasses the processors, memory, storage, networking hardware, and datacenter facilities that provide the complete execution environment, including emerging composable elements such as CXL-based memory pools [10].

This layered structure is intended to make explicit a claim that is often implicit in discussions of AI infrastructure: that the General-Purpose Compute layer is not a passive substrate beneath the "real" AI work happening in the Accelerator layer, but an active coordination layer whose efficiency directly determines whether the layers above and below it can operate at their full potential. A well-optimized Accelerator layer cannot compensate for an Orchestration layer starved of scheduling responsiveness, nor can efficient

scheduling compensate for a Physical Infrastructure layer that cannot supply memory fast enough to keep pace with a long-running agentic session. Table 1 (below) summarizes representative infrastructure demands introduced by Agentic AI relative to traditional single-turn inference workloads, and Table 2 (Section 4) maps specific architectural techniques discussed in Sections 2 and 3 to the compute-stack layer each technique primarily strengthens.

Table 1. Infrastructure Demand Comparison: Single-Turn Inference vs. Agentic Workloads

Dimension	Single-Turn Inference Workload	Agentic Workload
Request structure	Single forward pass through the model	Variable-length chain of model calls, tool invocations, and memory operations
Latency profile	Bounded by model compute time	Compounded across multiple dependent steps; control-plane latency becomes significant
Memory demand	Fixed by context window and batch size	Grows with session length and number of concurrent chained sub-calls
Scheduling assumption	Requests largely independent, similarly sized	Requests form dependency graphs; sub-task size often unknown until runtime
Dominant bottleneck risk	Accelerator throughput	Coordination across scheduling, memory, and orchestration layers

The stack is deliberately conceptual rather than a prescriptive reference architecture, since implementations will vary across cloud providers and deployment contexts. Its purpose is to provide a shared vocabulary for reasoning about where a given infrastructure investment or research contribution is targeted, and to make visible the cross-layer dependencies that a narrowly accelerator-focused view of AI infrastructure tends to obscure.

Viewed through this framework, several of the systems discussed in Sections 2 and 3 can be located precisely. PagedAttention and S3 primarily strengthen the boundary between the General-Purpose Compute and Accelerator layers, since both improve how efficiently scheduling and memory-management decisions translate into usable accelerator capacity: PagedAttention reports 2 to 4 times higher throughput than prior serving systems such as FasterTransformer and Orca at comparable latency, while S3 reports up to 6.49 times higher throughput than worst-case memory-reservation baselines by scheduling on predicted output length [12][13]. Fiddler and dLoRA operate at the same boundary but from the opposite direction, using general-purpose compute resources to compensate for accelerator memory constraints rather than the reverse: Fiddler achieves roughly an order-of-magnitude speedup (approximately 8 to 10 times) over offloading baselines for uncompressed mixture-of-experts inference on memory-constrained GPUs, and under its evaluated LoRA-serving workloads derived from real-world request traces, dLoRA reports up to 57.9× and 26.0× higher throughput than vLLM and HuggingFace PEFT, respectively[14][15]. CXL-based memory pooling operates primarily at the Physical Infrastructure layer but has direct consequences for the General-Purpose Compute layer's ability to support long-running or memory-intensive agentic sessions [9][10]. Agentic memory systems and RAG architectures, by contrast, operate mainly within the AI Orchestration layer, though as noted above they generate substantial downstream demand on the General-Purpose Compute layer for indexing, retrieval, and storage coordination [7][22]. Mapping techniques to layers in this way is useful precisely because it clarifies that no single layer's optimization is sufficient on its own: a fully optimized Accelerator layer paired with an unoptimized Orchestration layer will still produce a system that performs poorly under real agentic workloads, and the reverse is equally true.

Table 2. Architectural Techniques Mapped to the Agentic AI Compute Stack

Technique	Primary Mechanism	Compute-Stack Layer Strengthened
PagedAttention (vLLM)	Paged KV-cache memory management (2-4x throughput vs. prior serving systems)	General-Purpose Compute / Accelerator boundary
S3	Output-length-aware scheduling (up to 6.49x throughput vs. worst-case reservation)	General-Purpose Compute / Accelerator boundary
Fiddler	CPU-assisted MoE inference orchestration (~8-10x speedup vs. offloading baselines)	General-Purpose Compute / Accelerator boundary
dLoRA	Dynamic adapter migration across replicas (up to 57.9x throughput vs. vLLM)	General-Purpose Compute / Accelerator boundary
CXL memory pooling	Composable, cross-server memory expansion	Physical Infrastructure
NUMA-aware scheduling	Locality-preserving placement	General-Purpose Compute
RAG / agentic memory systems	External knowledge retrieval and dynamic memory organization	AI Orchestration

Table 3 situates the Agentic AI Compute Stack relative to four existing architectural perspectives, clarifying what each already addresses well and where the unified framework extends beyond it.

Table 3. Comparison of the Agentic AI Compute Stack with Existing Architectural Perspectives

Architectural Perspective	Primary Focus	Typical Unit of Analysis	What It Does Not Explicitly Address	How the Agentic AI Compute Stack Extends It
Cloud computing / virtualization literature [17][18][19]	Elastic provisioning and multi-tenant isolation of general-purpose compute	The virtual machine or container instance	How compute is coordinated across an agentic request's chained, dependent steps	Treats virtualization as one layer whose scheduling decisions must be coordinated with orchestration, accelerator, and memory layers rather than optimized in isolation
LLM inference-serving systems research [12][13][14][15]	Maximizing throughput and memory efficiency for a single model-serving deployment	The individual inference request or batch	Multi-step, cross-service execution spanning tool calls, retrieval, and multi-agent coordination	Locates serving-layer techniques (e.g., PagedAttention, Fiddler) at a specific General-Purpose Compute / Accelerator boundary within a larger five-layer execution path
Agent frameworks and multi-agent orchestration research [1][3][5][6][8]	Reasoning, planning, and inter-agent communication patterns	The agent or the multi-agent conversation	The underlying infrastructure demands (scheduling, memory pooling, virtualization) that	Situates agent frameworks within the AI Orchestration layer and traces their downstream

			these reasoning patterns place on cloud compute	resource demands onto the General-Purpose Compute and Physical Infrastructure layers
Composable hardware / CXL literature [9][10]	Physical-interconnect-level memory and resource pooling across servers	The server or memory pool	How pooled resources are actually scheduled and allocated to specific agentic workloads	Positions composable hardware as the Physical Infrastructure layer feeding into General-Purpose Compute scheduling decisions, rather than a standalone hardware concern

5. FUTURE DIRECTIONS

Several trends are likely to shape the continued evolution of general-purpose cloud compute for Agentic AI. Composable infrastructure enabled by CXL is expected to mature substantially, allowing compute and memory resources to be allocated dynamically according to workload requirements rather than fixed at server-provisioning time [9][10]. As agentic sessions grow longer and multi-agent deployments grow more concurrent, this flexibility is likely to shift from a performance optimization to a baseline architectural requirement [10].

AI-assisted infrastructure management represents a second significant direction. Machine-learning-centric resource management research has already demonstrated the feasibility of learned scheduling and capacity-planning policies that outperform static heuristics under variable load [16][20]; extending these techniques specifically to the branching, dependency-laden execution patterns of agentic workloads is a natural next step, and one that several of the serving-systems innovations discussed in Section 2, such as dynamic adapter migration and output-length-aware scheduling, already anticipate in narrower form [13][15].

Multi-agent orchestration itself continues to evolve rapidly. Recent surveys of LLM-based multi-agent frameworks document substantial variation across current systems in state-management granularity, token-cost structure, and failure-recovery design, indicating that orchestration-layer standardization remains an open problem rather than a solved one [8]. This lack of standardization has direct infrastructure consequences: a cloud provider supporting several different orchestration frameworks simultaneously must accommodate meaningfully different assumptions about how state is persisted, how failures propagate, and how token consumption is tracked, each of which shapes the demands placed on the underlying General-Purpose Compute layer. Related work cataloging currently deployed agentic systems similarly finds that documentation of safety practices, formal evaluation, and governance remains inconsistent across the field, suggesting that infrastructure-level standardization may need to proceed alongside, rather than after, governance and safety standardization [3]. Infrastructure providers are therefore likely to face pressure to support observability and auditing capabilities, such as tracing a multi-step agentic execution across services, tools, and memory operations, as a first-class infrastructure concern rather than an afterthought, since governance and safety commitments are difficult to substantiate without the underlying compute layer being able to reconstruct what an agent actually did at each step.

Sustainability considerations are also likely to grow in importance as agentic workloads scale. Because agentic sessions consume compute across multiple coordination steps rather than a single inference pass, their aggregate energy footprint is more difficult to characterize using inference-only benchmarks, and energy-efficiency research in cloud computing will likely need to account explicitly for orchestration overhead rather than accelerator utilization alone [21][20]. Hardware-accelerator research aimed specifically at large language model workloads is progressing quickly [11], but as this paper has argued throughout, accelerator-level

efficiency gains will only translate into system-level sustainability gains if the coordination layers surrounding those accelerators are optimized in tandem.

A related priority is the development of evaluation metrics that measure efficiency across a complete agentic workflow rather than at the level of an individual inference request or a single accelerator's utilization. Metrics such as per-request latency or per-GPU utilization, however useful for conventional inference serving, do not capture the cumulative scheduling, memory-locality, and cross-service coordination overhead that accumulates across a multi-step agentic execution graph. Future infrastructure research would benefit from workload-level measures, for example, end-to-end coordination overhead as a share of total session time, or memory-locality penalty accumulated per agentic session, that treat the full chain of planning, tool invocation, and memory operations as the unit of analysis, rather than any single step within it. Establishing such metrics is a prerequisite for the kind of empirical benchmarking called for elsewhere in this paper, since coordination-layer overhead cannot be optimized systematically until it can first be measured at the workflow level.

Table 4. Qualitative Directional Shift in Dominant Infrastructure Coordination Concern

Phase (approximate)	Dominant Infrastructure Coordination Concern (per literature reviewed)	Representative Technologies
2010-2016	Multi-tenant virtualization and elastic resource provisioning for enterprise workloads	Hypervisor-based virtualization; specialized VM families for compute-, memory-, storage-, and network-intensive workloads [17][18][19]
2017-2022	Offline training throughput, followed by early online-inference latency and reliability demands	Batch-oriented training infrastructure; early online-serving stacks addressing inference latency and availability [10][12][20][21]
2023-present	Inference-serving memory/scheduling efficiency (PagedAttention, S3) and adapter/expert-routing coordination (Fiddler, dLoRA) under agentic, multi-step workloads	PagedAttention (vLLM), S3, Fiddler, dLoRA; CXL-based composable memory pooling; NUMA-aware and topology-aware schedulers [9][10][12][13][14][15]

Future research should therefore prioritize integrated optimization across compute, networking, storage, memory, and accelerators, rather than optimizing any single layer of the Agentic AI Compute Stack in isolation. This is likely to require closer collaboration between the systems-research community, which has driven recent serving-layer innovations, and the agentic-AI research community, which continues to drive rapid change in the workload patterns that this infrastructure must support. A useful indicator of progress on this front will be the extent to which future serving-systems research explicitly characterizes agentic, multi-step workloads as a distinct benchmarking category, rather than continuing to evaluate new techniques primarily against single-turn inference baselines, since the latter risks understating the coordination overhead that dominates real agentic deployments.

6. DISCUSSION AND CONCLUSION

This paper has argued that the scalability of Agentic AI increasingly depends on addressing architectural coordination bottlenecks alongside accelerator-performance constraints, particularly as agentic workloads grow in complexity, duration, and concurrency. This position does not diminish the importance of GPU and accelerator innovation; rather, it contends that accelerator gains are necessary but insufficient, because the orchestration, scheduling, memory-management, and multi-agent-coordination demands introduced by Agentic AI are handled predominantly by general-purpose compute, and inefficiency at that layer constrains overall system performance regardless of how capable the underlying accelerator is [8][12]. The literature reviewed in Sections 2 and 3 supports this position across several independent lines of systems research: inference-serving innovations such as PagedAttention, S3, Fiddler, and dLoRA each address a distinct coordination bottleneck (memory paging, GPU utilization under variable output length, CPU-GPU data movement, and multi-adapter serving, respectively), none of which is resolved by faster accelerator hardware alone [12][13][14][15].

The Agentic AI Compute Stack introduced in Section 4 synthesizes these findings into a structure intended to be useful beyond this paper: it offers infrastructure researchers and cloud architects a shared frame for locating where a given technique or investment is targeted, and for identifying cross-layer dependencies that a narrower, accelerator-centric framing would leave implicit. A reasonable objection to this framing is that accelerator performance remains the ultimate ceiling on what any coordination layer can enable, since no amount of scheduling sophistication can substitute for insufficient raw compute. This paper's position is not that accelerator performance is unimportant, but that it is not sufficient on its own, and that the marginal returns to further accelerator-only investment are likely to diminish relative to the returns available from coordinated cross-layer optimization, particularly as agentic workloads grow in length, concurrency, and inter-agent complexity.

A second reasonable objection concerns the conceptual, rather than empirically benchmarked, nature of the Agentic AI Compute Stack itself. Because the framework synthesizes findings from separately reported systems rather than presenting new head-to-head measurements, it does not itself quantify how much of a given deployment's latency or cost is attributable to coordination-layer inefficiency versus accelerator constraints in any specific production environment. This is an appropriate limitation to state plainly: the framework's contribution is to provide a vocabulary and a mapping, of the kind illustrated in Section 4, for organizing the increasingly fragmented body of infrastructure research relevant to Agentic AI, not to supply a validated performance model. Establishing quantitative benchmarks that isolate coordination-layer overhead from accelerator-layer overhead across representative agentic workloads is precisely the kind of empirical follow-on work this paper's synthesis is intended to motivate.

As AI systems continue to move toward greater autonomy, cloud providers will need to optimize complete execution pipelines rather than individual hardware components in isolation. The evidence reviewed here, spanning agentic reasoning frameworks, inference-serving systems research, agent memory architectures, and composable infrastructure technologies, indicates that this coordinated, cross-layer optimization is already underway across multiple independent research communities, even where it has not yet been synthesized into a single architectural narrative. This paper's contribution has been to provide that synthesis, and to argue that future infrastructure research and investment should treat general-purpose compute as a first-order determinant of Agentic AI scalability, not a secondary concern subordinate to accelerator design. Looking ahead, the Agentic AI Compute Stack is intended to serve as a conceptual foundation rather than a finished result: a shared frame on which future empirical benchmarking of coordination-layer overhead, agent-aware scheduling policies, composable-memory research extending beyond CXL, and workflow-level infrastructure optimization can build. As agentic workloads continue to grow in scale, autonomy, and inter-agent complexity, the infrastructure community's ability to reason precisely about where a given bottleneck resides, and to develop the corresponding empirical tools to measure it, will depend on treating the full stack, and not any single layer within it, as the proper unit of analysis.

ETHICS STATEMENT

Conflict of Interest: The author declares no conflict of interest.

Statement of Human and Animal Rights: This article does not contain any studies involving human participants or animals performed by the author. This is a conceptual and literature-synthesis paper.

Informed Consent: Not applicable, as this study did not involve human participants.

AUTHOR CONTRIBUTION

Priyadarshni Shanmugavadivelu is the sole author of this manuscript and is responsible for conceptualization, literature review, analysis, framework development, writing, and final approval of the submitted version.

DATA AVAILABILITY STATEMENT

This is a conceptual and literature-synthesis paper; no new empirical datasets were generated or analyzed. All sources synthesized are cited in the numbered reference list.

BIOGRAPHICAL SKETCH

Priyadarshni Shanmugavadivelu is a cloud infrastructure and AI systems professional with extensive industry experience in general-purpose compute platform architecture, large-scale distributed systems, and cloud infrastructure strategy for AI workloads. Her professional work has focused on the architectural coordination challenges that arise when scaling compute infrastructure to support demanding, latency-sensitive AI

applications, an area of practical experience that directly informs this manuscript's central argument regarding the importance of general-purpose compute in enabling scalable Agentic AI systems.

REFERENCES

[1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. 11th Int. Conf. Learning Representations (ICLR), Kigali, Rwanda, 2023. https://arxiv.org/abs/2210.03629

[2] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, 2023. https://arxiv.org/abs/2302.04761

[3] S. Casper, L. Bailey, R. Hunter, C. Ezell, E. Cabale, M. Gerovitch, S. Slocum, K. Wei, N. Jurkovic, A. Khan, P. J. K. Christoffersen, A. P. Ozisik, R. Trivedi, D. Hadfield-Menell, and N. Kolt, "The AI agent index," arXiv:2502.01635, 2025. https://arxiv.org/abs/2502.01635

[4] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[5] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, 2023. https://arxiv.org/abs/2303.11366

[6] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv:2308.08155, 2023. https://arxiv.org/abs/2308.08155

[7] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," arXiv:2312.10997, 2023 (rev. 2024). https://arxiv.org/abs/2312.10997

[8] "LLM-based multi-agent orchestration: A survey of frameworks, communication protocols, and emerging patterns," Future Internet, MDPI, 2026. https://doi.org/10.3390/fi18060326

[9] D. Das Sharma, R. Blankenship, and D. S. Berger, "An introduction to the Compute Express Link (CXL) interconnect," ACM Computing Surveys, vol. 56, no. 11, Art. 290, 2024. https://doi.org/10.1145/3669900

[10] C. Chen, X. Zhao, G. Cheng, Y. Xu, S. Deng, and J. Yin, "Next-gen computing systems with Compute Express Link: A comprehensive survey," arXiv:2412.20249, 2024. https://arxiv.org/abs/2412.20249

[11] C. Kachris, "A survey on hardware accelerators for large language models," Applied Sciences, vol. 15, no. 2, p. 586, 2025. https://doi.org/10.3390/app15020586

[12] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with PagedAttention," in Proc. 29th ACM Symp. Operating Systems Principles (SOSP), 2023. https://doi.org/10.1145/3600006.3613165

[13] Y. Jin, C.-F. Wu, D. Brooks, and G.-Y. Wei, "S3: Increasing GPU utilization during generative inference for higher throughput," in Advances in Neural Information Processing Systems (NeurIPS), 2023. https://arxiv.org/abs/2306.06000

[14] K. Kamahori, Y. Gu, K. Zhu, and B. Kasikci, "Fiddler: CPU-GPU orchestration for fast inference of mixture-of-experts models," arXiv:2402.07033, 2024; in Proc. Int. Conf. Learning Representations (ICLR), 2025. https://arxiv.org/abs/2402.07033

[15] B. Wu, R. Zhu, Z. Zhang, P. Sun, X. Liu, and X. Jin, "dLoRA: Dynamically orchestrating requests and adapters for LoRA LLM serving," in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), Santa Clara, CA, 2024. https://www.usenix.org/conference/osdi24/presentation/wu-bingyang

[16] T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, "Machine learning (ML)-centric resource management in cloud computing: A review and future directions," Journal of Network and Computer Applications, vol. 204, Art. 103405, 2022. https://doi.org/10.1016/j.jnca.2022.103405

[17] L. A. Barroso, J. Clidaras, and U. Holzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 2nd ed. San Rafael, CA: Morgan & Claypool Publishers, 2013. https://doi.org/10.2200/S00516ED2V01Y201306CAC024

[18] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, pp. 50-58, 2010. https://doi.org/10.1145/1721654.1721672
[19] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST Special Publication 800-145, National Institute of Standards and Technology, 2011. https://doi.org/10.6028/NIST.SP.800-145
[20] Y. Sanjalawe, S. Al-E'mari, S. Fraihat, and S. Makhadmeh, "AI-driven job scheduling in cloud computing: A comprehensive review," Artificial Intelligence Review, 2025. https://doi.org/10.1007/s10462-025-11208-8
[21] R. Buyya, S. Ilager, and P. Arroba, "Energy-efficiency and sustainability in new generation cloud computing: A vision and directions for integrated management of data centre resources and workloads," Software: Practice and Experience, vol. 54, no. 1, pp. 24-38, 2024. https://doi.org/10.1002/spe.3248
[22] W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang, "A-MEM: Agentic memory for LLM agents," arXiv:2502.12110, 2025. https://arxiv.org/abs/2502.12110