



OPEN
Compute Project

AI Fabric Specialization for Agentic AI

From General-Purpose Inference to Autonomous, Long-Running Agent Workflows

Open Compute Project Foundation (OCP)
AI HW/SW Co-Design Workgroup
2026

Contributors:

- Weifeng Zhang, Lenovo Research & OCP AI HW/SW Co-Design
- Victor Lu, Independent
- Deep Shah, Socratic AI
- Prakash Jain, Celestica
- And members from OCP AI HW/SW Co-Design Workgroup

Executive Summary

Agentic AI has matured rapidly over the past two years, with autonomous systems that plan, reason, act, and coordinate over multi-turn, extended-horizon workflows now moving from research demos to production deployments. This shift from single-turn inference to multi-turn workflows places new demands on AI datacenter infrastructure. While this workgroup’s 2026 “AI Fabrics Perspective and Modeling” whitepaper (will be referred as **AFPM** onwards) established a solid framework for distributed training and stateless inference, agentic workloads introduce distinct requirements that the current framework does not fully address.

This paper addresses three of the six research directions identified by the workgroup in its 2026 proposal, “AI Fabric Specialization for Agentic AI”. These directions sit most directly at the intersection of agentic AI runtime behavior and AI fabric design: (1) Long-Context and Stateful Memory Management, and (2) Reliability, Fault Tolerance, and Recovery for Stateful Agent Tasks (3) Data Center backend network extension. All these directions are deeply connected to, but extend beyond, the distributed data services and checkpointing capabilities described in the AFPM whitepaper. The remaining directions will be further studied in the follow-up work.

The paper characterizes the infrastructure problems, surveys the current state of the industry, proposes concrete solutions including extensions to distributed storage services, new fabric-level primitives, and novel agent-aware protocols, and identifies gaps that represent open research challenges for the workgroup and the broader community.

Central contributions of this paper include: (1) a taxonomy of agentic memory tiers and their fabric implications, building on the KV-cache-centric disaggregated architectures pioneered by Mooncake and productized in NVIDIA Dynamo; (2) a proposal for a vendor-neutral Persistent Distributed KV Cache (PD-KVC) abstraction integrated with the AI Fabric via RDMA-class transports including UEC®1.0 and UALink®; (3) an Agent State Journaling (ASJ) model that applies the durable-execution / journal-based-replay paradigm, established in Temporal, AWS Lambda durable functions, Restate, and DBOS, to fabric-integrated, causal-aware checkpointing for agent workflows; (4) proposed extensions to the data center backend network (the AI Fabric) for agentic workloads, including a new latency-sensitive small-message traffic class and a Durable Write primitive; (5) extensions to the InfraGraph schema to represent session affinity, memory pools, and agent task graphs; and (6) a discussion of multi-agent coordination fabric primitives that complement application-layer protocols such as MCP and A2A. The paper concludes with an open challenges section, which identifies directions for future standardization and research.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Table of Contents

1. Introduction.....	4
2. Background: From Stateless Inference to Agentic AI.....	7
3. Research Direction 1: Long-Context and Stateful Memory Management.....	8
3.1 Problem Statement and Industry Context.....	8
3.2 Current Approaches and Their Limitations.....	10
3.3 Proposed Solutions.....	11
3.4 Gap and Future Work.....	15
4. Research Direction 2: Reliability, Fault Tolerance, and Recovery.....	17
4.1 Problem Statement and Industry Context.....	17
4.2 Current Approaches and Their Limitations.....	18
4.3 Proposed Solutions.....	19
4.4 Gaps and Future Work.....	22
5. Research Direction 3: Data Center Backend Networks Extension for Agentic AI.....	23
5.1 Problem Statement and Industry Context.....	23
5.2 Key Requirements for Enhanced Scale-Across Networks for Agentic AI.....	23
5.3 Current Landscape and Limitations.....	23
5.4 Proposed Solution: Scale-Across Distributed DC Architecture.....	24
5.5 Future Work and Evolution.....	25
6. Supporting Topics: Multi-Agent Coordination and Observability.....	26
6.1 Multi-Agent Coordination Infrastructure.....	26
6.2 Observability and Cost Attribution.....	27
7. InfraGraph Extensions for Agentic AI.....	29
8. Conclusion.....	31
9. Reference.....	33
Appendix A.....	36
Appendix B.....	47
Compliance with Open Compute Project Tenets.....	51
About Open Compute Project.....	52

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

1. Introduction

The 2026 “AI Fabrics Perspective and Modeling” whitepaper (**AFPM**) [1], published by the OCP AI HW/SW Co-Design Workgroup, established a foundational, vendor-neutral framework for understanding and designing AI datacenter infrastructure. That framework characterized AI workloads primarily through the lens of distributed training and single-turn inference, categorized network topologies into Scale-In, Scale-Up, Scale-Out, and Scale-Across domains, and introduced the InfraGraph modeling schema as a standardized tool for infrastructure description, simulation, and optimization.

While preparing that whitepaper, we observed a clear shift in AI deployment driven by the rapid rise of agentic AI. As illustrated in Figure 1, rather than responding to discrete user prompts in isolated, stateless exchanges, agentic systems autonomously plan sequences of actions, execute tool calls, retrieve and update memory across multiple reasoning steps, spawn and coordinate sub-agents, and operate over extended time horizons, often from minutes to hours or even days. Prominent examples include autonomous coding assistants such as Claude Code [38] and OpenAI Codex [39] that iteratively write, test, and debug software; research agents that retrieve documents, synthesize findings, and produce structured reports; enterprise automation agents that interact with APIs, databases, and communication systems; and multi-agent pipelines in which a planner orchestrates specialized executor, verifier, and critic agents.

The advancement of agentic AI imposes new infrastructure demands which go beyond the assumptions of the AFPM framework. Stateless inference assumes that each request begins with a cold context; agentic tasks demand persistent, shared, and efficiently retrievable state. Standard checkpointing was designed to recover model weights after training failures; agentic recovery must restore the causal history of a partially executed workflow, accounting for external side effects that cannot be replayed. Standard traffic management optimizes large, bursty collective communication patterns; agentic workloads generate frequent, low-latency, small-message roundtrips between model inference, tool execution backends, and memory stores. This distinction, recoverability of causal history rather than just state, is the same one that has driven the broader durable-execution movement in distributed systems over the past three years [25, 26], and we draw on that literature heavily in Section 4.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

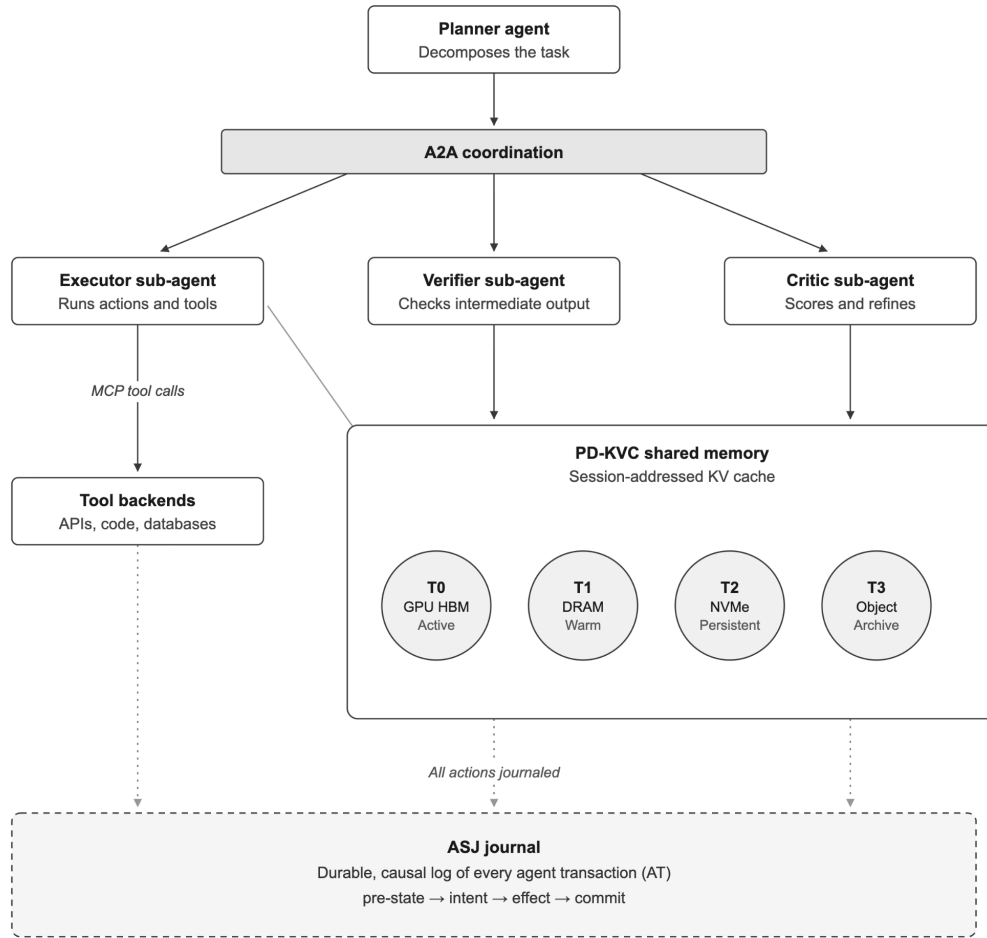


Figure 1. High-level abstraction of an agentic workflow. It shows a planner agent orchestrating multiple sub-agents (executor, verifier, critic), each interacting with tool execution backends via MCP and with shared KV cache state via PD-KVC. All agent actions are journaled via ASJ.

The 2026 Proposal of “AI Fabric Specialization for Agentic AI” [2], submitted to this workgroup, identified six major research directions arising from this agentic trend. Of these six, this paper addresses the three where agentic AI runtime behavior places the most direct demands on AI Fabric design:

- Long-Context and Stateful Memory Management (Section 3),
- Reliability, Fault Tolerance, and Recovery (Section 4), and

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

- Data Center Backend Networks Extension for Agentic AI (Section 5).

In addition, two supporting topics from the same six directions, Multi-Agent Coordination Infrastructure and Observability, are treated in Section 6 because they bear directly on the fabric designs proposed in the primary sections of this paper. Section 7 augments the InfraGraph schema with concrete extensions to capture the new primitives introduced in the paper.

This paper assumes the agent runtime and the inference fleet share a single operator-controlled fabric. The thin-client model, in which an agent calls a remote model API over the public internet and the serving side never sees agent identity, is out of scope: the fabric primitives proposed here (PD-KVC, ASJ, and the Agent Coordination Channel (ACC)) all depend on RDMA data paths, a shared journal pool, and session-tagged QoS enforced at the NIC and switch, none of which exist across that boundary.

Throughout, the paper draws on, extends, and in some cases re-adapts the concepts and terminology introduced in the AFPM whitepaper. Readers familiar with that document will find continuity in the treatment of distributed data services, RDMA integration, traffic management, and the InfraGraph schema, as well as new proposals beyond the AFPM framework.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

2. Background: From Stateless Inference to Agentic AI

As characterized in the AFPM whitepaper [1], traditional LLM inference is stateless by design. A user submits a prompt, the model performs a forward pass to generate a completion, and the session ends. During the prefill phase, the model computes attention keys and values for all input tokens in a single forward pass. During the decode (generation) phase, the model produces one token at a time; each step computes the key and value for the new token and, without caching, would require recomputing the keys and values of all prior tokens. The KV cache stores these key and value tensors so the decode phase can reuse them, eliminating most redundant computation.

Ephemeral by design in classical serving, the KV Cache resides in GPU HBM and is discarded at session termination. Until around 2024, the main goal of the infrastructure was throughput-centric: maximizing concurrent short-lived requests with low per-token latency while improving GPU utilization. The Mooncake architecture (powering Moonshot AI's Kimi service) and NVIDIA's Dynamo framework (GTC 2025) have since demonstrated that even non-agentic inference benefits from treating KV cache as a first-class, disaggregated resource. Agentic AI pushes this trend further and applies it to a wider class of workloads.

Agentic AI breaks the classical serving model along several dimensions:

- First, tasks are long-horizon. A single agent run may involve dozens to hundreds of sequential reasoning steps, tool invocations, and memory retrievals, with the total wall-clock duration spanning minutes to hours.
- Second, tasks are stateful. The agent must maintain coherent context across all steps, including its observations, prior actions, tool call outputs, and intermediate conclusions.
- Third, tasks often involve multi-agent coordination. Specialized sub-agents are spawned, executed concurrently or sequentially, and must share state efficiently.
- Fourth, tasks have external effects. Unlike inference which produces text, agents write files, call APIs, send messages, and modify databases. These actions cannot simply be retried if a failure occurs mid-task.

These properties create infrastructure requirements that span compute, memory, network, and storage layers and that differ in kind from those of training or stateless inference. The following sections analyze each of these research directions in detail, grounding the analysis in the infrastructure taxonomy and technology landscape established by the AFPM whitepaper.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

3. Research Direction 1: Long-Context and Stateful Memory Management

3.1 Problem Statement and Industry Context

The KV cache is the central memory primitive of modern transformer inference. During the prefill phase, the model processes all input tokens and computes key and value vectors for each attention head at each layer. Then, these KV vectors are stored in GPU HBM to avoid recomputation during the autoregressive decoding phase.

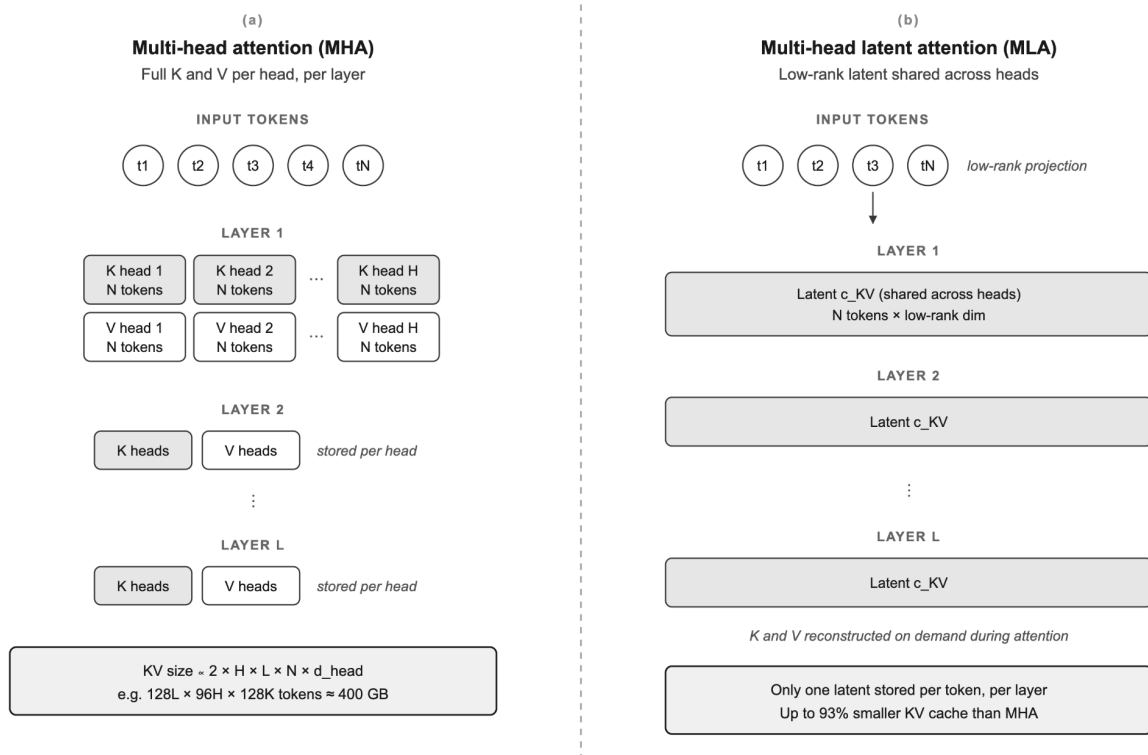


Figure 2. The KV Cache Architecture for transformer attention.

Left: classical MHA with separate K and V tensors stored per head per layer.

Right: MLA, which compresses K and V into a low-rank latent representation, reducing KV cache size by up to 93% relative to MHA.

For a multi-head attention (MHA) model with H attention heads, L layers, and a context length of N tokens, the KV cache occupies memory proportional to $2 \times H \times L \times N \times d_{\text{head}} \times \text{dtype_bytes}$. For example, a hypothetical 128-layer, 96-head MHA model with $d_{\text{head}}=128$,

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

serving a 128K-token context in FP16, would require ~825 GB of KV cache memory, already exceeding the HBM of a single high-end GPU.

This figure changes substantially for modern attention variants. Grouped-Query Attention (GQA), used in Llama 3 and Mistral families, reduces KV memory by 4-8× through K/V head sharing. Multi-head Latent Attention (MLA), introduced in DeepSeek-V2 and adopted across DeepSeek-V3, R1, and Kimi K2, compresses K and V into a low-rank latent representation and has been shown to reduce KV cache size by up to 93% relative to MHA at comparable or higher quality [19]. Frontier deployments in 2026 routinely target context windows of 1M tokens or more [43]. With MLA's compression, even a session that accumulates 1M tokens can fit within one high-capacity node's HBM. The fabric problem arises at the aggregate level: a cluster serving many concurrent long-context agent sessions, each persisting state across reasoning steps and sharing context across sub-agents, places total KV-cache demand well beyond any single Scale-Up domain's HBM. Compression changes the per-session arithmetic; the aggregate conclusion stands.

In agentic workloads, this problem is further compounded by three factors.

- The first is context accumulation: as an agent executes tool calls and incorporates their results, the effective context grows monotonically throughout the task. A coding agent working through a large codebase may accumulate hundreds of thousands of tokens of context across dozens of reasoning steps.
- The second factor is session persistence: unlike a stateless inference request, the agent's context must be preserved between reasoning steps, potentially across pauses, preemptions, or hardware failures. The context should be retrieved efficiently when the agent resumes.
- The third factor is multi-agent sharing: when a planner agent spawns executor sub-agents, those sub-agents may need to read portions of the planner's context, creating a shared memory problem that crosses both GPU and node boundaries.

Industry deployments have already run into these limitations. There are multiple approaches to address these limitations but each with its own pitfalls:

- A naive approach may try to keep the entire context in GPU HBM for the duration of an agent task, but it is prohibitively expensive at scale, wasting GPU memory capacity on idle context between reasoning steps and crowding out other inference requests.
- An alternative approach, sometimes called context stuffing, re-ingests the full context from CPU memory or storage at each reasoning step, but the quadratic attention complexity makes this latency-prohibitive for long contexts.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

- A third approach, context compression or summarization, reduces context length by replacing detailed history with compressed summaries, but degrades model performance on tasks requiring precise recall of earlier observations or tool call outputs.

The AFPM whitepaper anticipated this challenge in Section 5.A.d and introduced the concept of AI-native Inference Context Memory Storage: a cluster-level key-value cache extending GPU memory and enabling high-speed sharing of context across nodes for multi-turn, long-context agentic AI. However, the whitepaper treated this as a point feature of distributed storage. NVIDIA has since productized this concept: the Rubin platform introduced at CES 2026 includes an Inference Context Memory Storage platform for rack-scale KV cache offload [42], which confirms the industry trajectory while remaining vendor-specific. This section develops it into an architectural framework for stateful memory management across agentic workloads.

3.2 Current Approaches and Their Limitations

The industry has developed several partial solutions to the agentic memory problem, each with significant limitations.

A. Paged KV Caching (vLLM PagedAttention)

PagedAttention, introduced by vLLM, maps the KV cache to non-contiguous physical memory pages, enabling more efficient utilization of GPU HBM by eliminating internal fragmentation and supporting dynamic sequence length growth. This improves GPU memory utilization for serving multiple concurrent requests, but does not address the fundamental capacity problem for very long contexts, nor does it provide any persistence across session boundaries or sharing across agent instances. The KV cache remains entirely within a single GPU node's memory.

B. KV Cache Offloading to CPU Memory or NVMe

Systems such as FlexGen and others have explored offloading portions of the KV cache to CPU DRAM or even NVMe storage when GPU HBM is exhausted, retrieving cache entries as needed during decoding. While this extends effective context capacity, the PCIe® bandwidth bottleneck between GPU and CPU memory (~64 GB/s per direction on PCIe 5.0 x16, ~128 GB/s aggregate) creates severe latency penalties for attention computation that must access offloaded cache blocks frequently. Since KV offload traffic is dominated by reads back into the GPU during decode, the per-direction figure is the binding constraint. At 128K token context lengths, even modest attention head counts make PCIe offloading impractical for latency-sensitive agentic reasoning steps.

C. RAG and External Vector Stores

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Retrieval-Augmented Generation (RAG) allows models to retrieve relevant context chunks from external databases using embedding-based similarity search, rather than attending over the full context. For certain workload types, such as knowledge retrieval tasks in particular, RAG can effectively substitute for long-context attention. However, RAG is not a general solution for agentic state management: it cannot faithfully represent ordered action histories, tool call outputs with precise reference semantics, or the causal dependencies between reasoning steps that make agent tasks coherent. RAG retrieval also introduces additional round-trips to vector store services, adding latency on the critical path of each reasoning step.

D. Emerging Inference Disaggregation (Prefill-Decode, P/D Disaggregation)

Inference disaggregation is a mechanism to separate prefill computation (which is compute bound) and decode computation (which is memory bandwidth bound) on different GPU pools, connected by high-speed fabrics. Recent industry work has productionized disaggregated prefill/decode (P/D) serving. NVIDIA Dynamo (announced at GTC 2025) provides a Kubernetes-native distributed inference framework with disaggregated prefill/decode, LLM-aware request routing, multi-tier KV cache management with S3/Azure-blob storage offload, and a Planner that dynamically rebalances prefill and decode workers based on real-time signals [24]. The Mooncake platform that serves Moonshot AI's Kimi service introduced this disaggregation pattern in production a year earlier and reports 75% more requests served compared to its prior vLLM-based stack on real workloads [20]. The SGLang, vLLM v1, and TensorRT-LLM communities have all integrated the Mooncake Transfer Engine as a KV connector backend during 2025. NIXL (NVIDIA's inference transfer library) supports Mooncake as a plug-in backend.

These deployments validate the core thesis of this paper, that KV cache is a first-class, fabric-managed resource, but they remain (a) tightly coupled to specific inference stacks, (b) optimized for stateless or short-session workloads rather than the long-horizon, multi-agent patterns that dominate agentic AI, and (c) lacking standardization across vendors. PD-KVC, as proposed below, attempts to abstract these systems into a vendor-neutral, agent-aware reference architecture suitable for inclusion in InfraGraph.

3.3 Proposed Solutions

A. Tiered Agentic Memory Architecture

We propose a four-tier memory architecture for agentic workloads, designed to match the access latency and bandwidth requirements of different memory types to the capabilities of different storage and network technologies. This architecture extends the distributed data services framework of the AFPM whitepaper with agent-specific semantics.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Tier	Storage Medium	Latency Target	Bandwidth	Contents
T0: Active KV	GPU HBM	~ns to sub- μ s	TB/s (HBM-resident)	Current step KV cache
T1: Warm KV Pool	RDMA-accessible DRAM cluster pool	1-10 μ s intra-PoD; 5-20 μ s cross-PoD	100s GB/s per GPU	Recent steps, shared agent context
T2: Persistent KV Store	NVMe SSD via GPUDirect Storage	read 10-100 μ s; write 50-200 μ s (sustained)	read 10s GB/s per GPU; write a few GB/s per GPU sustained	Full session history, cross-session, HAC snapshots
T3: Archive / Episodic Memory	Distributed object store (S3-compat.)	ms to s	1-10 GB/s	Long-term agent memory, audit log

Table 1. Four-tier memory architecture for agentic workloads.

Here, Tier 0 (T0) corresponds to the conventional GPU HBM KV cache, used for the active reasoning step. Tier 1 (T1) is the new primary contribution: a cluster-level DRAM pool, accessible via RDMA over the Scale-Up or Scale-Out fabric, serving as a high-speed backing store for the KV cache entries of recent reasoning steps and for shared context across co-located agent instances. Tier 2 (T2) provides persistent storage via NVMe SSDs, accessed through GPUDirect Storage to bypass CPU overhead, supporting full session persistence and cross-session retrieval. Tier 3 (T3) provides long-term episodic memory storage in a distributed object store, suitable for audit logging, agent memory replay, and cross-run learning.

B. Persistent Distributed KV Cache (PD-KVC) Architecture

The Persistent Distributed KV Cache (PD-KVC) is not, in itself, a novel storage architecture. Production systems including Mooncake Store, NVIDIA Dynamo’s KV-aware routing, LMCache, TokenLake, KVFlow, MemServe, and FlexKV all implement variations of distributed KV caching today. PD-KVC’s contribution is a vendor-neutral, agent-aware reference architecture defined in terms that InfraGraph can model and that the OCP community can converge on. PD-KVC implements the T1 and T2 tiers described in Table 1 above as an integrated service within the AI Fabric. We design PD-KVC around the following principles:

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

- **Session-Addressed Namespace:** Each agent task is assigned a globally unique session identifier. KV cache entries are addressed by a compound key (*session_id*, *layer_id*, *token_range*), enabling fine-grained retrieval of specific attention layers for specific token ranges without loading the full context. This is distinct from the block-addressed page tables used by PagedAttention, which are local to a single GPU node, and complementary to the segment-level addressing recently proposed in TokenLake. Mooncake Store's prefix-hash-based addressing operates at a similar level of abstraction, but PD-KVC additionally couples session identifiers to agent task identifiers (Section 7) to enable fabric-level QoS attribution. These keys address entries within a fixed model, precision, and kernel context; entries produced under a different configuration are not interchangeable and are namespaced separately. To preserve cross-session reuse, PD-KVC supports a two-level addressing scheme: immutable shared prefixes (for example a common system prompt) are content-hash addressed and deduplicated across sessions, while the mutable, agent-specific remainder is session-scoped. This keeps deduplication for shared context without weakening per-session isolation or QoS attribution.
- **RDMA-Native Data Path:** PD-KVC exposes an RDMA-accessible interface so the NIC or DPU transfers KV cache entries directly between the cluster-level DRAM pool and GPU memory, keeping the CPU off the data path. In the default path a host thread still posts the work request; GPU-initiated RDMA (for example IBGDA) removes the CPU from the control path as well. The AFPM whitepaper demonstrated that RDMA integration via GPUDirect Storage sharply reduces latency and CPU overhead for storage access; PD-KVC applies the same principle to the KV cache layer, treating attention memory as a managed communication flow rather than a local resource.
- **Staging and Prefetch:** Decode is memory-bandwidth-bound and offers almost no compute to overlap a transfer against, so the KV cache for the active reasoning step must already be resident in T0 HBM before decode begins rather than demand-paged during it. PD-KVC stages a session's working set from T1 or T2 into HBM ahead of its next step, following the prestaging model used by Mooncake and NVIDIA Dynamo rather than faulting on the decode critical path. Demand-paging and prefetch apply at session transitions: when a suspended session resumes, or when a planner hands off to a sub-agent, the PD-KVC controller issues non-blocking RDMA reads to pull the required KV ranges back into HBM before the resuming step runs. A prefetch engine, informed by the agent's execution trace and the *AgentTaskGraph* of Section 7, anticipates which session will run next and overlaps its staging with the current step's compute. This staging-ahead parallels the asynchronous I/O prefetching described for training data loading in Section 5.3 of the AFPM whitepaper.
- **Multi-Agent Shared Context:** When a planner agent spawns sub-agents that require access to a shared context prefix, PD-KVC enables zero-copy sharing: sub-agents

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

receive a reference to the shared session namespace rather than a copy of the KV cache entries. This eliminates the $O(N \times \text{num_agents})$ memory overhead of context duplication and reduces the fabric traffic required for multi-agent workflows. Access control is enforced at the session namespace level, preventing unauthorized access across tenants. KVFlow has shown that workflow-aware KV management, tracking each agent's steps-to-execution in an Agent Step Graph and prioritizing retention accordingly, can meaningfully reduce cache misses in multi-agent workflows. PD-KVC adopts this notion as a direct scheduling input via the *AgentTaskGraph* nodes proposed in Section 7. This connects the KV-residency layer (PD-KVC) to the workload characterization layer (InfraGraph) and enables the Mapping Decision Engine of the AFPM whitepaper to reason about agentic placement directly.

- **Eviction and Lifecycle Management:** The PD-KVC controller implements a session-aware eviction policy, distinguishing between active sessions (where eviction would cause stalls), suspended sessions (where eviction to T2 is acceptable), and completed sessions (where entries may be archived to T3 or discarded). Eviction policies are configurable per-tenant and can incorporate priority signals from the cluster orchestrator, enabling QoS-aware memory management across multiple concurrent agent tasks. Because the multi-agent sharing described above lets several sessions reference the same KV blocks, eviction is governed by reference counting: a shared block becomes eligible for eviction or archival only after the last referencing session releases it. Unshared, session-private entries follow the per-session active, suspended, and completed states directly, while shared prefixes persist until no live session depends on them.

C. Fabric Requirements for PD-KVC

The PD-KVC architecture places specific requirements on the AI Fabric that extend the framework of the AFPM whitepaper. The Scale-Up fabric, which the AFPM document defined as supporting sub-microsecond latency with memory semantics within a PoD, and which is now instantiated by UALink (1.0 ratified April 2025 at 200G/lane and up to 1024 accelerators per pod; 2.0 ratified April 2026, adding In-Network Compute) and proprietary alternatives such as NVLink® 6, introduced with the Vera Rubin platform in 2026, is the primary target for T1 reads and writes. The bandwidth requirements for PD-KVC T1 operations are determined by the KV cache access pattern during attention computation. Layers execute sequentially, and each decode step reads the KV cache of every layer once to produce a single token. Because a step completes in milliseconds and repeats continuously, the sustained read bandwidth for a 128K token context can reach hundreds of gigabytes per second per GPU, placing PD-KVC in the Scale-Up bandwidth regime.

For multi-agent shared context that spans PoD boundaries, the Scale-Out fabric, instantiated by UEC 1.0 which defines AI Full and AI Base profiles explicitly tuned to small-message AI traffic,

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

provides the transport for T1 reads from remote DRAM pools. This introduces a latency asymmetry in multi-agent workflows: intra-PoD sub-agents benefit from Scale-Up latency, while cross-PoD coordination incurs Scale-Out latency. Intelligent agent placement, which co-locates collaborating agent instances within the same PoD whenever possible, becomes a critical optimization. This suggests a new dimension for the workload-to-hardware mapping problem described in Section 7 of the AFPM whitepaper.

The PD-KVC system also motivates a new traffic class in AI Fabric QoS management. The AFPM whitepaper distinguished synchronous, latency-sensitive traffic (Tensor Parallelism, Pipeline Parallelism, real-time inference) from asynchronous, throughput-sensitive traffic (Data Parallelism All-Reduce, checkpointing, data loading). PD-KVC T1 reads fall into a third category: latency-sensitive but small-message, occurring at high frequency on the critical path of agentic reasoning steps. This class requires low-latency scheduling and priority treatment in the fabric, analogous to the RDMA Read operations used in high-performance distributed databases, but at much higher frequency and smaller granularity. Because these small reads share scale-up links with per-layer Tensor Parallelism all-reduce and, across PoDs, scale-out links with bulk KV and snapshot transfers, they are exposed to incast: many small, high-frequency requests converging on shared buffers alongside large flows. Isolating this class from bulk traffic, through dedicated queues or priority, is therefore a requirement, but its behavior under sustained congestion is not yet characterized.

The T0 tier itself must be shared safely when multiple agent tenants are co-located on the same accelerator. GPU memory and compute isolation at the orchestration layer is provided today by Kubernetes-native device-virtualization middleware such as HAMi (Heterogeneous AI Computing Virtualization Middleware) [37], a CNCF Sandbox project that enforces per-workload device-memory and compute limits and partitions a single physical GPU across pods without application changes. HAMi governs how the HBM-resident T0 tier is divided among co-located tenants, while PD-KVC governs the T1-T3 tiers across the fabric. HAMi's scheduler-extender and device-plugin model is a candidate enforcement point for the per-session memory quotas of Section 4.3.D and the session-aware placement layer of Section 3.4. At the agent runtime layer above, Kubernetes-native frameworks such as kagent (CNCF Sandbox) [40] already represent agents as custom resources, which makes per-agent session identifiers visible to the orchestrator and simplifies the mapping from agent sessions to fabric-level quotas.

3.4 Gap and Future Work

Several important challenges remain open for the PD-KVC architecture and the broader agentic memory management problem.

- **Attention Algorithm Co-Design:** Current transformer attention algorithms assume fully resident KV caches in GPU HBM. Extending algorithms, such as FlashAttention to

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

support paged, RDMA-accessible KV caches without sacrificing the fused kernel properties for efficiency, is a non-trivial research problem which requires co-design between AI Fabric architects and ML system researchers.

- **Compression and Quantization for T1/T2:** The bandwidth demands of PD-KVC T1 can be reduced significantly by applying quantization or compression to KV cache entries before storing them in the DRAM pool. Techniques such as KV cache quantization to INT4 or INT8 can reduce transfer volume by 4-8x. Integrating hardware-accelerated compression in the NIC or DPU (as proposed for checkpointing compression in Section 5.D of the AFPM whitepaper) would make this transparent to the GPU compute path. Independently of compression, coherent memory-pooling fabrics such as CXL® 4.0 (PCIe 7.0, 128 GT/s, multi-rack pooling) [34] offer a complementary substrate for the T1 warm-pool tier where rack-scale DRAM sharing is available.
- **Session Routing and Placement Algorithms:** The efficiency gains from PD-KVC depend on placing agent instances close to their KV cache in the fabric topology. This requires a session-aware scheduling layer in the cluster orchestrator that tracks KV cache residency across the memory tier hierarchy and routes new reasoning steps to GPU instances within the same PoD as the cached state. Extending the InfraGraph schema to represent session affinity constraints and KV cache occupancy is a prerequisite for such algorithms (see Section 7).
- **Attention Mechanisms Beyond Transformers:** As the industry adopts architectures with fundamentally different state management (e.g., state-space models (SSMs) and hybrid architectures), the specific memory primitives of PD-KVC may need generalization. The framework of tiered, RDMA-accessible agent memory remains applicable, but the specific cache structure will vary.

Transfer versus Recompute for Suspended Sessions: PD-KVC migrates KV cache bytes between tiers, but a token log is several orders of magnitude smaller than the KV cache it produces. For sessions resuming after a long pause, re-running prefill from the stored token log may cost less than transferring KV entries back from T1 or T2. Whether transfer or recompute wins depends on context length, the per-direction bandwidth of the tier holding the cache, and whether the prefetch engine can hide transfer latency behind compute. A policy that chooses per session, informed by the AgentTaskGraph nodes of Section 7, is an open design question.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

4. Research Direction 2: Reliability, Fault Tolerance, and Recovery

4.1 Problem Statement and Industry Context

In traditional AI training, a failure mid-run means lost computation since the last checkpoint. The cost is internal: wasted GPU-hours and delayed convergence, with no external consequences. The job is simply restarted from the checkpoint. In agentic AI, the failure calculus changes. An agent that was 80% complete on a complex task may have already sent emails, committed code, called external APIs, modified database records, or triggered financial transactions. A naive restart from a system checkpoint would repeat those actions, causing duplicate or inconsistent side effects, producing an incomplete and/or incorrect result. This is the failure model that the durable execution programming paradigm has formalized over the past three years in the broader distributed-systems community. The agentic AI community has begun to converge on the same paradigm, LangGraph's checkpointing, Microsoft Agent Framework's Checkpoint Manager, OpenAI Agents SDK's durable-execution integration, Pydantic AI's Temporal binding, and the NVIDIA-initiated Dapr Agents framework are all evidence of this convergence, but adoption remains fragmented and operates entirely above the infrastructure layer. The proposals in this section aim to close that gap by integrating durable execution with the AI Fabric.

The industry has begun to recognize this distinction, but infrastructure-level solutions remain immature. Early agentic deployments, including enterprise automation systems and autonomous coding assistants, have reported high failure rates in complex multi-step tasks, with failures arising from model errors, tool call timeouts, network partitions, and GPU hardware failures. Without infrastructure-level support for graceful recovery, operators have been forced to implement application-level retry logic, manual intervention protocols, or simply truncate task complexity to reduce failure probability.

The AFPM whitepaper addressed fault tolerance primarily in the context of distributed training, where the recovery model is well-defined: detect node failures, reload the last checkpoint, and restart training with the surviving nodes, adjusting the parallelism strategy as needed. Section 5.F described the checkpointing infrastructure in detail, and Section 5.1.7 outlined the fabric-level requirements for failure detection, node isolation, and topology readjustment. This infrastructure provides a valuable foundation for agentic fault tolerance, but the agentic recovery model differs in critical ways that require new primitives.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

4.2 Current Approaches and Their Limitations

A. Application-Level Checkpointing

The most common current approach to agentic fault tolerance is application-level checkpointing, in which the agent framework periodically serializes the agent's state, i.e., its context window, tool call history, and memory, to a durable store such as a database or object storage service. While effective for simple failure modes, this approach has several limitations.

- First, it is coarse-grained: checkpoints are typically taken at step boundaries, so a failure mid-step loses the partial work of that step and requires re-execution.
- Second, it is not causally aware: the checkpoint records what state the agent was in, but not which external effects it has already produced. A recovery that re-executes from the checkpoint without this causal information may produce duplicate or even incorrect external effects.
- Third, it is implemented above the infrastructure layer, without coordination with the AI Fabric or distributed data services, so recovery cannot draw on fabric-level failure detection or topology reconfiguration.

B. Tool Call Idempotency

A complementary approach is to design agent tool interfaces to be idempotent, meaning that repeated calls with the same arguments produce the same result without additional side effects. Idempotency simplifies recovery by allowing re-execution without worrying about duplicate effects.

However, idempotency is difficult or impossible to guarantee for many real-world tool categories: financial transactions, email sending, file system modifications in concurrent environments, and stateful API calls are inherently non-idempotent. Idempotency remains a useful design principle, but it cannot cover these tool categories.

C. Saga Pattern and Compensating Transactions

Distributed systems engineering has long addressed similar problems in the context of distributed transactions, producing patterns such as the Saga pattern, in which long-running transactions are decomposed into sequences of smaller, individually committed steps, each with an associated compensating action that can undo its effects if a later step fails. Applying the Saga pattern to agentic workflows is promising but requires formalization of agent actions as transactions with explicit compensating actions. This is a significant challenge for autonomous agents with open-ended, LLM-generated action sequences. Some agentic frameworks have begun exploring lightweight Saga-inspired recovery, but without infrastructure-level support from the AI Fabric or data services layer.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

4.3 Proposed Solutions

A. Agent State Journaling (ASJ)

We propose ASJ as a fine-grained fabric-integrated realization of the durable-execution paradigm for agentic AI. Extending the checkpointing infrastructure described in the AFPM whitepaper with a causally-aware mechanism, ASJ borrows semantics directly drawn from write-ahead logging in distributed databases and from the journal-and-replay model used by Temporal and Restate, while adding two elements absent from current application-layer durable execution: tight coupling with the AI Fabric's RDMA, QoS, and failure-detection layers, and explicit integration with PD-KVC tier migration as part of the recovery state. ASJ defines the fabric primitives that runtimes such as LangGraph, Temporal, and DBOS can target.

The core abstraction of ASJ is the Agent Transaction (AT), which captures a single atomic unit of agent progress. An AT consists of four components:

- 1) A pre-state snapshot, representing the agent's full logical state (context, memory references, tool call history) immediately before the action;
- 2) An intent record, describing the action about to be taken (tool call, memory write, sub-agent spawn) with all parameters;
- 3) An effect record, appended after the action completes, recording the actual output, any external side effects produced, and a completion status; and
- 4) A commit marker, written only after the AI Fabric confirms that the effect record has durably persisted. This structure mirrors a write-ahead log record in a distributed database: the journal is the source of truth for the agent's causal history.

These four components map directly to the journal-event structure used in Temporal's history, the step records in AWS Lambda Durable Functions, and the event-sourced state in DBOS. Section 4.4 discusses standardization of the AT schema across runtimes.

Recovery from a failure proceeds by reading the journal from the last committed AT. The recovery agent can determine exactly which actions were completed and their results, which actions were in-flight at the time of failure and require compensating actions or re-execution, and which actions had not yet started and can be safely re-executed. This provides the causal awareness missing from coarse-grained application-level checkpointing.

B. Fabric Integration of ASJ

For ASJ to be practical, journal writers must not become a bottleneck on the critical path of agentic reasoning. The AFPM whitepaper demonstrated that high-bandwidth parallel

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

checkpointing, using RDMA and GPUDirect Storage, can reduce checkpoint write times for multi-terabyte training states to minutes. For ASJ, the requirements are different: journal entries are small (kilobytes to megabytes per AT), but writes must be low-latency and durable before the agent proceeds to the next action. These are write-ahead log semantics rather than bulk checkpoint semantics.

We propose that ASJ journal writes be integrated into the AI Fabric as a distinct traffic operation, using RDMA Write with Immediate operations to a dedicated journal storage pool with persistence guarantees. The journal storage pool is implemented on high-endurance NVMe SSDs with hardware-accelerated write journaling (similar to the Flash Tier described in Section 5.H.a of the AFPM whitepaper), connected to the AI Fabric via a high-radix, low-latency path. The NIC or DPU on the compute node performs the journal write in parallel with the tool call execution, so that by the time the tool call completes, the intent record is already durably committed.

This integration requires a new primitive in the AI Fabric: a Durable Write confirmation signal. Where a standard RDMA Write completion confirms only that data was delivered to the remote node, a Durable Write confirmation signals that the journal write was committed to persistent storage on the journal node. It is a durability acknowledgment, closer in spirit to an fsync or commit acknowledgment in a storage system than to a transport-level signal.

The fabric must confirm that the journal write was committed to persistent storage on the journal node, which goes beyond confirming transmission. This is a stronger guarantee than what current RDMA Write semantics provide, and implementing it efficiently requires co-design between the NIC, the switch fabric, and the storage backend. Such cross-layer co-design is exactly the kind of work the OCP workgroup is positioned to drive. UEC 1.0's Reliable Ordered Delivery (ROD) and Reliable Unordered Delivery (RUD) modes provide foundations on which such a primitive could be standardized, potentially as a new opcode class in the UET Semantic Sub-layer. We recommend that the OCP and UEC workgroups jointly evaluate this in their next coordination cycle.

C. Hierarchical Agent Checkpointing (HAC)

Not all agent states require journal-level durability at every step. We propose a HAC framework that combines ASJ fine-grained journaling with periodic coarse-grained snapshots, analogous to the combination of write-ahead logging and periodic full checkpoints in database systems.

Under HAC, the agent runtime takes a full snapshot of agent state at configurable intervals, typically at natural task boundaries such as the completion of a major subtask or the handoff between sub-agents. The agent state may include the full KV cache (migrated to T2 persistent storage via PD-KVC), the agent's memory references, the complete tool call history, and all environment variables. Between full snapshots, ASJ provides fine-grained recovery to the exact AT boundary. This combination allows recovery latency to be bounded: in the worst case,

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

recovery requires replaying ATs since the last full snapshot, which can be controlled by snapshot frequency.

The full snapshot mechanism builds directly on the distributed checkpointing infrastructure described in the AFPM whitepaper. KV cache snapshots are written to T2 persistent storage via GPUDirect Storage and the AI Fabric, using the same high-bandwidth parallel write path designed for training checkpoints. The key difference is that agent state snapshots are typically much smaller than training checkpoints (gigabytes rather than terabytes) but must be written more frequently (every few minutes rather than every few hours), so the performance requirements shift from peak bandwidth to sustained IOPS and write latency.

D. Runaway Agent Detection and Circuit Breaking

A distinct failure mode specific to agentic AI is the runaway agent: an agent that enters an unproductive loop, consumes excessive resources, or takes unintended external actions due to model errors or adversarial inputs. The 2026 proposal identified this as a critical safety and reliability concern, asking what mechanisms are needed to halt or contain runaway agents without disrupting other agents or enterprise systems.

We propose a fabric-level Circuit Breaker mechanism for agentic workloads. The circuit breaker monitors three signals:

- 1) Resource consumption (GPU time, memory usage, fabric bandwidth, tool call count) against per-session budgets defined by the orchestrator;
- 2) Semantic anomaly detection, in which a lightweight monitoring model analyzes the agent's action trajectory and flags statistically unusual patterns; and
- 3) External watchdog signals from tool execution backends, which can report unexpected call patterns or policy violations.

These signals add a fabric-level layer beneath the application-level guardrails proposed in the AI Runtime Infrastructure framework [35], which remain in place. Recent work on canonical-path-deviation monitoring has shown that flagging the bottom tercile of agent runs by trajectory adherence at 75% completion and triggering a controlled restart can lift success rates by ~8.8 percentage points among the intervened runs, suggesting that fabric-level circuit breakers integrating such signals can deliver measurable reliability gains across tenants.

When the circuit breaker triggers, it initiates a controlled shutdown sequence: new tool calls from the agent are blocked at the fabric level, the current reasoning step is allowed to complete (to avoid leaving tool calls in an inconsistent mid-execution state), the agent's ASJ journal is flushed and sealed, and the orchestrator is notified with a structured incident report including the full causal history. This sequence is analogous to the failure isolation and topology readjustment

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

described in Section 5.1.7 of the AFPM whitepaper, adapted from hardware node failure to application-level agent misbehavior.

The fabric-level implementation of the circuit breaker benefits from the AI Fabric's ability to enforce flow control and QoS policies at the NIC and switch level, as described in Section 3.5 of the AFPM whitepaper. By tagging all agent traffic with session identifiers and enforcing per-session resource quotas in the fabric, the circuit breaker can block specific sessions without affecting other tenants. This is a capability that pure application-level enforcement cannot guarantee if the agent's host process is compromised or unresponsive.

4.4 Gaps and Future Work

- **Formal Verification of Recovery Protocols:** The correctness of ASJ recovery depends on precise semantics for AT commit and effect recording. Formally specifying and verifying these semantics, particularly in the presence of partial failures during the journal write itself, requires techniques from distributed systems formal methods that have not yet been applied to AI Fabric design. The workgroup should engage with the database and distributed systems research communities to establish formal correctness criteria.
- **Standardization of Agent Transaction Interfaces:** ASJ requires agent frameworks (LangChain, AutoGen, OpenAI Agents SDK, Microsoft Agent Framework, Dapr Agents, CrewAI, Google ADK, Pydantic AI, kagent, and other custom orchestrators) to expose structured intent and effect records. The OCP workgroup should engage with the Linux Foundation (which now stewards MCP through the Agentic AI Foundation (AAIF) and A2A through its own governance body) to propose a standard Agent Transaction Schema, since MCP itself does not specify journaling semantics.
- **Non-Idempotent Tool Call Recovery:** Even with ASJ, recovering from a failure that occurred after an external effect was produced but before the effect record was committed remains an open problem. Solving this requires coordination with external systems (e.g., email servers, databases, financial APIs) that may not support transaction rollback. The workgroup should evaluate a lightweight two-phase commit protocol between ASJ and external tool backends for high-value, non-idempotent actions.
- **Performance Characterization:** The latency overhead of ASJ journal writes on the critical path of agentic reasoning steps has not yet been measured in production deployments. Simulation using tools like ASTRA-sim, extended with ASJ workload traces, would provide early estimates. The InfraGraph schema extensions proposed in Section 7 are designed to enable such simulations.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

5. Research Direction 3: Data Center Backend Networks Extension for Agentic AI

5.1 Problem Statement and Industry Context

Agentic AI, characterized by persistent state management, complex inference, and multi-agent coordination, requires autonomous, low-latency edge decision-making. Thus, agentic AI not only necessitates moving compute resources to the network edge but also demands an integrated “across the domain” strategy. Data center (DC) AI Fabric backend networks, introduced in the AFPM whitepaper [1], utilize three scaling domains: Scale-Up (inter-PoD), Scale-Out (inter-PoDs), and Scale-Across (across distributed DCs/PoDs). While existing OCP frameworks manage local scale-up and scale-out environments effectively, current DC hyperscaler-centric network infrastructure is not sufficient to support the low-latency decision-making required by these edge-centric requirements. Hence, backend networks must be extended to the edge, possibly over Wide Area Networks (WAN) and Radio Access Networks (RAN). However, the primary technical hurdle involves maintaining rigorous performance standards as AI fabrics scale across WAN and RAN.

5.2 Key Requirements for Enhanced Scale-Across Networks for Agentic AI

To support distributed Agentic AI workloads across the WAN/RAN domains, backend networks must meet the additional requirements (in addition to those listed in [1], such as port bandwidth, sub-microsecond latency, predictability and near lossless communication) as follows:

- **Scale-Across Distributed Heterogeneous Hardware:** High-performance workloads distributed across various geographic locations and heterogeneous hardware.
- **Edge Proximity:** Compute deployment closer to end-users to support real-time inference and autonomous decision-making.
- **Hybrid Memory Access Models:** Integration for both Load/Store (memory-like semantics) at the edge and RDMA over WAN/RAN.

5.3 Current Landscape and Limitations

A. Cross-Industry Efforts

Significant innovation is underway across multiple industry consortia and organizations to address the bottlenecks of interconnect bandwidth and reliability.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

- **OCP Workstreams:** Ethernet Scale-Up Network (ESUN) optimizes framing and headers; Scalable Ultra Ethernet Transport (SUE-T) focuses on XPU co-design for low-latency memory transactions.
- **Complementary Initiatives:** UEC for RDMA-optimized transport; UALink for low-latency accelerator communication; and Spectrum Ethernet for deterministic cluster connectivity.

B. Hardware Innovations

Hardware vendors are simultaneously innovating system design and silicon capabilities to meet the stringent demands of AI infrastructure:

- Next-gen accelerators (e.g., NVIDIA Vera Rubin).
- High-performance networking ASICs (e.g., Broadcom Tomahawk Ultra1) with deep buffering and In-Network Compute capabilities.
- Advancements in Co-Packaged Optics (CPO) and high-density pluggable.

However, current approaches are primarily limited to hyperscaler data centers, insufficient to satisfy agentic AI's distributed, edge-centric requirements. Hence, a new solution is required.

5.4 Proposed Solution: Scale-Across Distributed DC Architecture

The scale-across architecture facilitates the operation of multiple distributed AI fabrics or PODs as a single, logically unified fabric. Key features of this architecture include a global substrate that extends backend fabrics across WAN and RAN environments, closer to edge/users, surpassing current restrictions that limit these fabrics to the interior of data centers. The proposed solution is built upon several core components designed to bridge these gaps (Red blocks in Figure 3):

- **Distributed Rack/Fabric Architecture:** Dis-aggregated compute for dynamic resource pooling across locations.
- **Control & Management Plane:** A unified control plane supporting heterogeneous fabrics, telemetry-driven optimization, and cross-domain resource scheduling.
- **Agentic Management:** Hierarchical agents implementing closed-loop control (Telemetry → Decision → Action).
- **Inter-Domain Coordination:** A hybrid architecture combining tightly and loosely coupled across the domain coordination.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

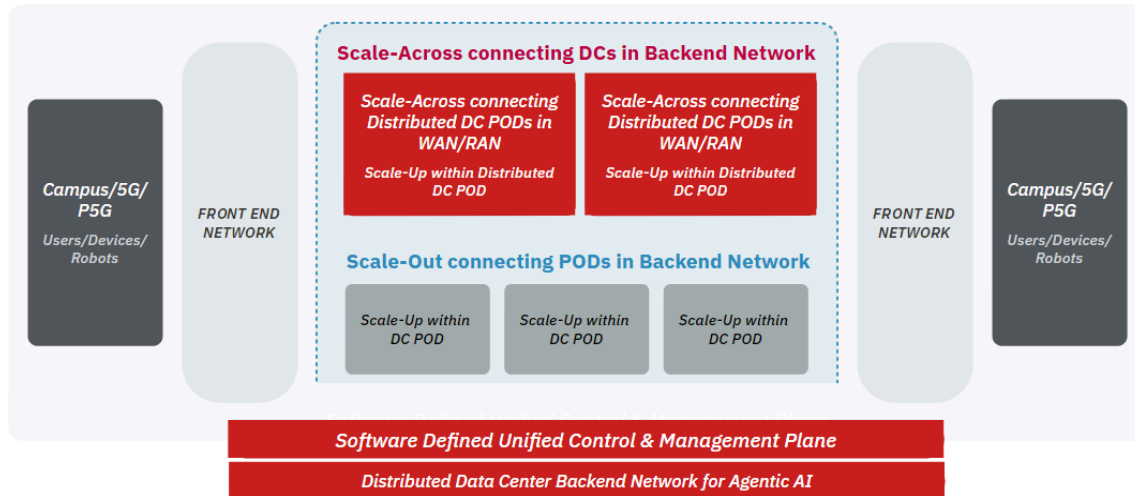


Figure 3. Scale-Across Distributed DC Fabric Architecture (Red colored blocks are the new components).

5.5 Future Work and Evolution

Scale-across backend networking represents a fundamental evolution of AI infrastructure, from isolated, high-performance fabrics to globally coordinated, distributed AI systems. This transition extends AI infrastructure from core data centers to the far edge and RAN deployments. Future advancements will focus on:

- **Silicon Photonics:** Reducing power with scalable optical interconnects and 3D photonic integration (targeting 100x bandwidth gains).
- **Transport Protocols:** Continued optimization of Ethernet-based fabrics.
- **Hardware Design:** Adoption of Open Rack Wide (ORW) and standardized chiplet architectures.
- **Precision Intelligence:** Implementing hierarchical management agents and picosecond-level synchronization in 6G RAN to ensure seamless, global coordination of agentic AI systems.

Ultimately, scale-across will enable agentic AI systems to operate seamlessly across distributed environments. By bringing compute closer to users at the edge while maintaining performance, these networks support the real-time, autonomous decision-making required for next-generation AI workloads.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

6. Supporting Topics: Multi-Agent Coordination and Observability

6.1 Multi-Agent Coordination Infrastructure

The first two of the primary research directions treated in this paper, stateful memory management and fault-tolerant recovery, are further complicated when multiple agents must coordinate. A planner agent that orchestrates a network of executor sub-agents requires persistence and recoverability for each agent's state, and additionally requires that the coordination state stay consistent across the multi-agent system. This includes which sub-agents have been assigned which tasks, what results they have reported, and what the current overall task graph looks like.

Current multi-agent systems rely primarily on application-layer messaging. The Model Context Protocol (MCP), originally published by Anthropic in November 2024 and donated to the Linux Foundation's Agentic AI Foundation in December 2025, has become the de facto standard for agent-to-tool communication and reports 97M+ monthly SDK downloads across its Python and TypeScript SDKs as of December 2025 [31]. The Agent-to-Agent (A2A) protocol, originated by Google, donated to the Linux Foundation in June 2025, and merged with IBM's Agent Communication Protocol (ACP) under the Linux Foundation in August 2025, addresses agent-to-agent coordination [32]. MCP and A2A define the application layer for agentic systems, but neither specifies the underlying transport guarantees, QoS treatment, or fabric-level traffic class. Data-plane proxies such as agentgateway (donated to the Linux Foundation in August 2025, now an Agentic AI Foundation project) [41] already mediate MCP and A2A traffic for security, observability, and governance, and represent a natural enforcement point between application-layer protocols and the fabric. The Agent Coordination Channel proposed below provides the fabric-layer transport that both A2A's application-layer coordination and gateway-mediated agent traffic depend on.

We propose that the AI Fabric should provide a dedicated Agent Coordination Channel (ACC), a lightweight messaging fabric optimized for the small-message, low-latency, ordered communication patterns of multi-agent coordination. The ACC is implemented as a separate QoS traffic class in the fabric, distinct from the data-parallel collective communication classes described in the AFPM whitepaper. It provides reliable, ordered delivery of small messages between agent instances. Because agent reasoning steps take seconds and coordination messages that are journaled as ATs (Section 4.3) carry durable-write overhead of tens of microseconds, the binding requirement is ordering and durability rather than minimal transport latency: journaling cost must stay small relative to step duration. Lightweight control signals that

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

do not require journaling, such as cancellation, can use the lower Scale-Up and Scale-Out latencies directly.

The ACC integrates with ASJ: agent coordination messages (task assignments, result reports, cancellation signals) are journaled as ATs, alongside tool call records, enabling the recovery mechanism to reconstruct the full multi-agent coordination state from the journal. This integration ensures that recovery after a failure is consistent across all agents in a multi-agent workflow, covering both each agent's local state and the coordination state between them.

6.2 Observability and Cost Attribution

The 2026 research proposal identified observability and cost attribution as a major challenge for agentic AI: in a single-turn chat product, cost is measured in tokens in plus tokens out; in an agentic system, cost is a complex function of model calls, tool executions, memory retrievals, sub-agent spawnings, and fabric operations spread over time and across services. Debugging why an agent behaved unexpectedly requires replaying a long causal chain across multiple systems.

The ASJ journal described in Section 4.3 provides the raw material for agentic observability: a complete, causal, durable record of every action the agent took, every tool call it made, and every external effect it produced. Building an observability service on top of ASJ enables several capabilities:

- End-to-end tracing of agent workflows, much like distributed tracing in microservice architectures but operating at the AI Fabric level, allows operators to visualize the full causal graph of an agent run and identify latency bottlenecks, failure points, and resource inefficiencies.
- Fine-grained cost attribution, computed by annotating each AT with the GPU time, fabric bandwidth, memory consumption, and tool call cost associated with it, provides the multi-dimensional cost breakdown that agentic billing requires.
- Causal replay, in which the ASJ journal is replayed in simulation against the InfraGraph model of the infrastructure, allows operators and researchers to understand agent behavior post-hoc and to test recovery procedures without affecting production systems.

The Digital Twin concept introduced in Section 6.5 of the AFPM whitepaper is a natural substrate for agentic observability. An Operational Digital Twin integrates the InfraGraph model with real-time telemetry to enable predictive maintenance and online simulation. By feeding ASJ records into the Digital Twin in real time, the orchestrator can monitor the resource consumption and behavioral patterns of all active agent sessions, detect anomalies consistent with runaway

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

agent behavior, and proactively reallocate KV cache memory tier assignments to prevent eviction-induced latency spikes before they occur.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

7. InfraGraph Extensions for Agentic AI

The InfraGraph schema described in Section 6 of the AFPM whitepaper provides a standardized, machine-readable representation of AI infrastructure as a graph of devices, components, links, and edges annotated with performance characteristics. This section proposes concrete extensions to InfraGraph to support the agentic AI concepts introduced in this paper. These extensions are designed to be backward-compatible with the existing schema and to enable the simulation, optimization, and operational monitoring use cases described above.

A. Memory Tier Nodes

The four memory tiers of the PD-KVC architecture (Table 1) should be represented as native node types in InfraGraph, alongside the existing GPU, CPU, NIC, and Switch node types. A MemoryTierNode carries attributes including tier level (T0-T3), capacity in bytes, access latency (mean and tail), bandwidth to the attached fabric interface, and eviction policy parameters. Edges between MemoryTierNodes and GPU nodes carry PD-KVC semantic annotations: RDMA access type (Read, Write, Atomic), bandwidth guarantee, and latency SLA.

B. Session Affinity Annotations

Edges and nodes in InfraGraph should support session affinity annotations, which indicate that a given agent session's KV cache is currently resident at a specific MemoryTierNode, and that GPU nodes processing that session's reasoning steps should be scheduled within the same PoD as that MemoryTierNode to minimize fabric hop count and latency. The orchestrator reads these annotations to make placement decisions and updates them as KV cache entries are migrated between tiers.

C. Agent Task Graph Nodes

The workload characterization dimension of InfraGraph (described in Section 7.3 of the AFPM whitepaper as a requirement for the Mapping Decision Engine) should be extended with an AgentTaskGraph node type. An AgentTaskGraph node represents a running or planned agentic workflow, with attributes including the session identifier, the current step index, the estimated remaining context growth rate, the tool call frequency, and the sub-agent dependency graph. This information enables the orchestrator to model the future resource demands of each agent session and to make proactive scheduling, placement, and tier migration decisions.

D. ASJ Journal Nodes

The ASJ journal storage pool should be represented in InfraGraph as a JournalNode, with attributes including write throughput (IOPS), write latency (mean and P99), durability guarantee (fsync or equivalent), and current occupancy. Edges from GPU nodes to JournalNodes carry

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

ASJ traffic class annotations, enabling the fabric traffic management layer to assign appropriate QoS priority to journal write operations.

E. Circuit Breaker Policy Edges

Circuit breaker policies (Section 4.3.D) should be represented as annotated policy edges between AgentTaskGraph nodes and the GPU and NIC nodes processing that agent's traffic. A CircuitBreakerEdge carries attributes including resource budget thresholds, watchdog timeout values, and the list of fabric-level enforcement actions (session throttle, session block, orchestrator notification) to be taken when a threshold is exceeded.

With these extensions, InfraGraph moves beyond static infrastructure description. It becomes a live operational model of the agentic AI fabric that captures the physical infrastructure alongside the dynamic state of running agent sessions, their memory tier assignments, their ASJ journal occupancy, and their circuit breaker status. This enriched model enables the Digital Twin to simulate the impact of adding a new high-priority agent session on the memory tier utilization of existing sessions, to optimize tier migration schedules to minimize eviction-induced latency, and to predict circuit breaker trigger events before they occur.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

8. Conclusion

This paper has presented a framework for extending AI Fabric infrastructure to meet the demands of agentic AI workloads. Beginning from the foundation established by the AFPM whitepaper, we have analyzed two primary research directions, Long-Context and Stateful Memory Management, and Reliability, Fault Tolerance, and Recovery, and proposed concrete architectural solutions for each.

For stateful memory management, we proposed the Persistent Distributed KV Cache (PD-KVC) as a vendor-neutral reference architecture that consolidates lessons from Mooncake, NVIDIA Dynamo, LMCache, TokenLake, KVFlow, MemServe, and related production systems into an OCP-aligned, InfraGraph-modelable form. PD-KVC organizes a four-tier memory hierarchy extending from GPU HBM through RDMA-accessible cluster DRAM pools to persistent NVMe storage and long-term object stores. PD-KVC enables session persistence, zero-copy multi-agent context sharing, and staged movement of session working sets over the AI Fabric ahead of each reasoning step, addressing the mismatch between the ephemeral KV cache of stateless inference and the long-horizon, shared state requirements of agentic tasks. The architecture places new requirements on the Scale-Up fabric (in particular, a new small-message, latency-sensitive traffic class for T1 RDMA reads) and motivates new session-aware placement algorithms for the workload mapping framework.

For fault tolerance and recovery, we proposed Agent State Journaling (ASJ) as the fabric-integrated counterpart to the durable-execution paradigm now established in Temporal, AWS Lambda durable functions, Restate, DBOS, Cloudflare Workflows, and Inngest, and increasingly adopted by agent frameworks including LangGraph, OpenAI Agents SDK, Microsoft Agent Framework, and Dapr Agents. ASJ extends that paradigm with hardware-accelerated journal writes, fabric-level durable-write confirmation, and tight coupling to PD-KVC tier migration. ASJ provides the causal history necessary to distinguish between completed external effects and actions that must be re-executed or compensated during recovery, addressing the main limitation of existing coarse-grained checkpointing approaches for agentic workloads. We combined ASJ with Hierarchical Agent Checkpointing (HAC) to bound recovery latency and proposed a fabric-level Circuit Breaker mechanism for detecting and safely terminating runaway agents.

In order to address the distributed and edge-centric requirements of Agentic AI, we proposed extensions to the existing Data Center Backend Network architectures. These enhancements incorporate scale-across distributed heterogeneous hardware positioned at edge proximity, alongside the integration of hybrid memory access models.

The supporting topics of multi-agent coordination and observability led to proposals for an Agent Coordination Channel (ACC) as a dedicated fabric QoS traffic class, and for the use of the ASJ journal as the substrate for end-to-end agentic tracing, cost attribution, and causal replay. These

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

mechanisms connect the AFPM whitepaper’s Digital Twin concept to the operational reality of managing complex multi-agent AI workflows.

Finally, we proposed concrete extensions to the InfraGraph schema, including Memory Tier Nodes, Session Affinity Annotations, Agent Task Graph Nodes, ASJ Journal Nodes, and Circuit Breaker Policy Edges. These extensions allow InfraGraph to serve as an operational substrate for the agentic AI fabric, rather than only a design-time modeling tool.

The paper also identified several open challenges: attention algorithm co-design for RDMA-accessible KV caches, formal verification of ASJ recovery protocols, standardization of Agent Transaction interfaces, and placement algorithms that scale to clusters of thousands of devices. These challenges represent a research agenda for the OCP AI HW/SW Co-Design Workgroup and the broader AI infrastructure community. As agentic AI moves from research demonstrations to enterprise production deployments, the infrastructure layer will become a critical determinant of whether these systems can be operated reliably, at acceptable cost, and within safety constraints. The framework presented in this paper provides a starting point for the standardization work required to meet these challenges.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

9. Reference

1. OCP AI HW/SW Co-Design Workgroup. "AI Fabrics Perspective and Modeling." Open Compute Project Foundation, July 2026.
2. OCP AI HW/SW Co-Design Workgroup. "Proposal for AI Fabric Specialization for Agentic AI." Open Compute Project Foundation, 2026.
3. Kwon, W. et al. "Efficient Memory Management for Large Language Model Serving with Paged Attention." SOSP 2023.
4. Sheng, Y. et al. "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU." ICML 2023.
5. Lewis, P. et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." NeurIPS 2020.
6. Zhong, Y. et al. "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving." OSDI 2024.
7. Garcia-Molina, H., and Salem, K. "Sagas." ACM SIGMOD Record, 1987.
8. MLCommons. "Chakra: Enabling Innovation in the Machine Learning Infrastructure Stack." <https://mlcommons.org/working-groups/research/chakra/>
9. NVIDIA. "BlueField-4: The Processor Powering the Operating System of AI Factories." NVIDIA Blog, October 2025.
10. Dao, T. et al. "FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning." ICLR 2024.
11. Meta Engineering. "Scaling LLM Inference: Innovations in Tensor Parallelism, Context Parallelism, and Expert Parallelism." Meta Engineering Blog, October 2025.
12. Ultra Ethernet Consortium. "Ultra Ethernet Specification 1.0." June 11, 2025, <https://ultraethernet.org>
13. UALink Consortium. "UALink 200G 1.0 Specification." April 8, 2025; "UALink 2.0 Specification." April 7, 2026. <https://ualinkconsortium.org/specifications/>
14. InfraGraph. "Infrastructure Graph Modeling Schema." <https://infragraph.dev/>
15. Mooncake: <https://github.com/kvcache-ai/Mooncake>
16. Liu, Z. et al. "LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference.", October 2025. <https://arxiv.org/pdf/2510.09665>
17. Red Hat / IBM Research et al. "Ilm-d: Kubernetes-Native Distributed Inference." 2025. <https://ilm-d.ai/>
18. NVIDIA. "NIXL: NVIDIA Inference Transfer Library." 2025.
19. DeepSeek-AI. "DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model." arXiv:2405.04434, May 2024.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

20. Qin, R. et al. "Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving." ACM Transactions on Storage, November 2025. arXiv:2407.00079. <https://github.com/kvcache-ai/Mooncake>
21. Wu, B. et al. "TokenLake: A Unified Segment-level Prefix Cache Pool for Fine-grained Elastic Long-Context LLM Serving." arXiv:2508.17219, August 2025.
22. "KVFlow: Efficient Prefix Caching for Accelerating LLM-Based Multi-Agent Workflows." NeurIPS 2025.
23. Hu, C. et al. "MemServe: Context Caching for Disaggregated LLM Serving with Elastic Memory Pool." arXiv:2406.17565, June 2024.
24. NVIDIA. "Dynamo: A Datacenter-Scale Distributed Inference Serving Framework." Announced at GTC, March 2025. <https://github.com/ai-dynamo/dynamo>
25. Temporal Technologies. "Durable Execution Architecture." <https://temporal.io/>
26. AWS. "Lambda Durable Functions." Launched at re:Invent 2025, December 2025.
27. Restate. "Journal-Based Durable Execution Engine." <https://restate.dev/>
28. DBOS Inc. "Durable Execution for Crashproof AI Agents." 2025. <https://www.dbos.dev/>
29. Cloudflare. "Workflows: Durable Execution on Workers." General availability, 2025.
30. Inngest. "Durable Execution for AI Agents." 2025-2026. <https://www.inngest.com/>
31. Anthropic. "Model Context Protocol Specification." November 2024. Donated to Linux Foundation Agentic AI Foundation, December 2025.
32. Linux Foundation Agentic AI Foundation. "Agent-to-Agent (A2A) Protocol Specification, v1.0." March 2026. Originated by Google; merged with IBM ACP, August 2025.
33. Linux Foundation. "Agentic AI Foundation (AAIF) Charter." December 2025.
34. Compute Express Link Consortium. "CXL 4.0 Specification." November 18, 2025.
35. Cruz, C. "AI Runtime Infrastructure." arXiv:2603.00495, February 2026.
36. Lee, W. Y. "Capable but Unreliable: Canonical Path Deviation as a Causal Mechanism of Agent Failure in Long-Horizon Tasks." arXiv:2602.19008, February 2026.
37. Project-HAMi. "HAMi: Heterogeneous AI Computing Virtualization Middleware." CNCF Sandbox Project. <https://github.com/Project-HAMi/HAMi>
38. Anthropic. "Claude Code: Agentic Coding Tool." Product documentation, 2025. <https://www.anthropic.com/product/claude-code>
39. OpenAI. "Introducing Codex." May 2025. <https://openai.com/index/introducing-codex/>
40. Project kagent. "kagent: Cloud Native Agentic AI." CNCF Sandbox Project, accepted May 2025. <https://github.com/kagent-dev/kagent>
41. agentgateway. "Agentgateway: Agentic Proxy for AI Agents and MCP Servers." Agentic AI Foundation (Linux Foundation) project. <https://agentgateway.dev/>

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

42. NVIDIA. "NVIDIA Kicks Off the Next Generation of AI With Rubin." January 5, 2026. <https://nvidianews.nvidia.com/news/rubin-platform-ai-supercomputer>
43. Gemini Team Google, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context", <https://arxiv.org/abs/2403.05530>

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Appendix A

Glossary

A2A (Agent-to-Agent Protocol) Application-layer protocol for coordination between agents. Originated by Google, donated to the Linux Foundation in June 2025, merged with IBM's ACP in August 2025. A2A v1.0 was released in March 2026. Does not specify transport guarantees or fabric-level traffic classes; the ACC provides those.

AAIF (Agentic AI Foundation) A Linux Foundation body that stewards MCP and, through its own governance structure, A2A. The OCP workgroup proposes engaging AAIF to standardize the Agent Transaction Schema (Section 4.4).

ACC (Agent Coordination Channel) A dedicated QoS traffic class in the AI Fabric for multi-agent coordination messages. Provides reliable, ordered delivery of small messages between agent instances. Coordination events are journaled as Agent Transactions via ASJ, so recovery can reconstruct the full multi-agent coordination state.

Agentic AI AI systems that autonomously plan sequences of actions, invoke tools, manage memory across reasoning steps, spawn sub-agents, and operate over extended time horizons, from minutes to hours or days. Contrasted with stateless, single-turn inference.

agentgateway An Agentic AI Foundation (Linux Foundation) data-plane proxy that mediates MCP and A2A traffic for security, observability, and governance. Represents a natural enforcement point between application-layer protocols and the fabric.

AgentTaskGraph An InfraGraph node type introduced in this paper. Represents a running or planned agentic workflow with attributes: session identifier, current step index, context growth rate, tool call frequency, and sub-agent dependency graph. Used by the orchestrator for proactive placement, scheduling, and tier migration decisions.

Agent Transaction (AT) The atomic unit of ASJ. Each AT has four components: (1) a pre-state snapshot of the agent's full logical state before the action; (2) an intent record describing the action and its parameters; (3) an effect record appended after the action completes; and (4) a commit marker written only after the AI Fabric confirms durable persistence. Mirrors a write-ahead log record in a distributed database.

AI Fabric The network and interconnect infrastructure connecting GPUs, memory pools, storage, and NICs within an AI datacenter. Encompasses Scale-In, Scale-Up, Scale-Out, and Scale-Across domains as defined in the AFPM whitepaper.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

All-Reduce A collective communication operation used in distributed training where gradient updates from all workers are summed and the result is broadcast back to every worker. Throughput-sensitive and asynchronous; shares fabric links with PD-KVC T1 reads.

ASJ (Agent State Journaling) A fabric-integrated, causally-aware durable-execution mechanism for agentic AI. Extends the AFPM whitepaper’s checkpointing infrastructure with a write-ahead log of Agent Transactions, tight coupling to RDMA and QoS layers, and integration with PD-KVC tier migration. Enables recovery that distinguishes completed external effects from actions that must be re-executed or compensated.

ASTRA-sim A network simulation framework used to model AI fabric traffic patterns. Proposed as a tool for characterizing the latency overhead of ASJ journal writes before production deployment.

Autoregressive Decoding The decode phase of LLM inference: the model generates one token at a time, using the KV cache to avoid recomputing attention over all prior tokens at each step. Memory-bandwidth-bound.

Canonical Path Deviation A metric for detecting runaway agents. Measures how far an agent’s action trajectory deviates from a learned canonical path. Flagging the bottom tercile of runs at 75% completion and triggering a controlled restart has been shown to improve success rates by ~8.8 percentage points.

Causal Replay An observability capability built on ASJ: the journal is replayed in simulation against the InfraGraph model, allowing operators to understand agent behavior post-hoc and test recovery procedures without affecting production systems.

Causal History The ordered record of all actions an agent took, their results, and the external effects they produced. ASJ preserves causal history so that recovery can determine exactly which effects have already occurred and which have not.

Checkpointing Periodically saving system state to durable storage so that a job can be restarted from the checkpoint after a failure. In training, this saves model weights. In agentic AI, application-level checkpointing saves agent context but lacks causal awareness; ASJ and HAC extend it with causal records.

Circuit Breaker A fabric-level mechanism that monitors active agent sessions for resource overuse, semantic anomalies, and watchdog signals from tool backends. When a threshold is exceeded, it blocks new tool calls at the fabric layer, allows the current reasoning step to complete, flushes and seals the ASJ journal, and notifies the orchestrator. Represented in InfraGraph as CircuitBreakerEdge policy annotations.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

CNCF (Cloud Native Computing Foundation) Vendor-neutral foundation hosting open-source infrastructure projects. HAMi and kagent are CNCF Sandbox projects referenced in this paper.

Context Accumulation The monotonic growth of an agent's effective context as it executes tool calls and incorporates results. A coding agent may accumulate hundreds of thousands of tokens across dozens of reasoning steps.

Context Stuffing A naive approach that re-ingests the full context from CPU memory or storage at each reasoning step. Impractical at long context lengths due to the quadratic attention complexity of re-running prefill.

Context Window The maximum number of tokens a model can attend to at once. Frontier models in 2026 target 1M tokens or more. Long context windows amplify KV cache memory demands.

CXL (Compute Express Link) A coherent interconnect standard for memory pooling. CXL 4.0 (November 2025, PCIe 7.0, 128 GT/s) enables multi-rack DRAM sharing and is a candidate substrate for the PD-KVC T1 warm-pool tier.

Dapr Agents An agentic framework initiated at NVIDIA and maintained in the Dapr community that integrates durable-execution patterns. Evidence of convergence toward ASJ-compatible runtime behavior.

DBOS A durable-execution platform for AI agents whose event-sourced state model directly informs the AT schema used in ASJ.

d_head The dimension of each attention head's key and value vectors. Determines per-head memory footprint within the KV cache. A value of 128 is used in the paper's reference calculation (~825 GB for a 128-layer, 96-head MHA model at 128K tokens in FP16).

Digital Twin Introduced in the AFPM whitepaper. An operational model integrating InfraGraph with real-time telemetry for predictive maintenance and online simulation. Extended in this paper by feeding ASJ records into the twin so it can monitor all active agent sessions and detect runaway behavior.

DPU (Data Processing Unit) A programmable network processor (e.g., NVIDIA BlueField) that offloads data-path operations (RDMA transfers, compression, hardware-accelerated journal writes) from the host CPU.

DRAM Dynamic RAM. The T1 tier of PD-KVC uses cluster-level DRAM pools accessible via RDMA, providing 1–20 μ s latency and hundreds of GB/s bandwidth per GPU.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Durable Execution A programming paradigm in which the runtime journals intent and effect at each step and replays from the last committed point after a failure. Implemented by Temporal, AWS Lambda durable functions, Restate, DBOS, Cloudflare Workflows, and Inngest. ASJ extends this paradigm to the AI Fabric layer.

Durable Write A new fabric primitive proposed in this paper. A standard RDMA Write completion confirms delivery to the remote node. A Durable Write confirmation additionally signals that data was committed to persistent storage. Required for ASJ journal semantics. Proposed as a new opcode class in the UET Semantic Sub-layer, building on UEC 1.0 ROD/RUD delivery modes.

dtype / dtype_bytes The numerical precision used to store tensor values. FP16 (2 bytes) is the baseline for KV cache size calculations in this paper. INT4 (0.5 bytes) and INT8 (1 byte) quantization can reduce KV cache transfer volume by 4–8×.

FlashAttention A fused GPU kernel for efficient attention computation that avoids materializing the full attention matrix in HBM. Extending FlashAttention to work with paged, RDMA-accessible KV caches without losing its efficiency properties is identified as an open research problem.

FlexGen A system that offloads KV cache to CPU DRAM or NVMe when GPU HBM is exhausted. Demonstrates PCIe bandwidth as the binding bottleneck for offloaded KV reads; impractical for latency-sensitive agentic reasoning at long context lengths.

FlexKV A distributed KV cache system whose design informs PD-KVC.

FP16 16-bit floating point. Standard precision for KV cache storage in the paper’s reference calculations. Two bytes per value.

GQA (Grouped-Query Attention) An attention variant used in Llama-3 and Mistral families that reduces KV cache memory by 4–8× through K/V head sharing. An intermediate design between full MHA and the more aggressive MLA compression.

GPUDirect Storage NVIDIA technology enabling GPUs to read from and write to NVMe storage directly, bypassing the CPU data path. Used in PD-KVC for T2 reads and writes, and in HAC for full snapshot operations.

GTC NVIDIA GPU Technology Conference. NVIDIA Dynamo was announced at GTC March 2025.

HAC (Hierarchical Agent Checkpointing) A checkpointing strategy combining fine-grained ASJ journaling with periodic full snapshots of agent state. Analogous to write-ahead logging plus

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

periodic full checkpoints in databases. Bounds recovery latency to the cost of replaying ATs since the last full snapshot.

HAMi (Heterogeneous AI Computing Virtualization Middleware) A CNCF Sandbox project that enforces per-workload GPU memory and compute limits, partitioning a single physical GPU across pods without application changes. Governs the T0 HBM tier for co-located agent tenants. Its scheduler-extender and device-plugin model is a candidate enforcement point for per-session memory quotas.

HBM (High Bandwidth Memory) GPU-attached DRAM with very high bandwidth (TB/s range). Stores the active KV cache (T0 tier) during inference. Capacity on a single high-end GPU is insufficient for the aggregate KV demands of many concurrent long-context agent sessions.

IBGDA (InfiniBand GPUDirect Async) A GPU-initiated RDMA mechanism by which the GPU directly posts RDMA work requests, removing the CPU from the control path entirely. Referenced as an advanced PD-KVC data path option.

Idempotency A property of tool calls: repeated calls with the same arguments produce the same result without additional side effects. Simplifies recovery but cannot be guaranteed for financial transactions, email, or stateful API calls.

In-Network Compute Computation performed by switch ASICs or smart NICs rather than host CPUs or GPUs. Added to UALink 2.0 (April 2026). Enables collective operations to be offloaded to the fabric.

InfraGraph A vendor-neutral, machine-readable schema for describing AI infrastructure as a graph of devices, links, and performance annotations. Introduced in the AFPM whitepaper. This paper extends it with MemoryTierNodes, Session Affinity Annotations, AgentTaskGraph Nodes, JournalNodes, and CircuitBreakerEdges.

Incast A network congestion condition in which many senders simultaneously target a single receiver, causing buffer overflow. Relevant to PD-KVC T1 reads, where many small, high-frequency requests share scale-up links with large bulk transfers.

IOPS (Input/Output Operations Per Second) A storage performance metric. ASJ journal writes require sustained IOPS rather than peak bandwidth, since entries are small (KB to MB) but written frequently on the critical path of agent reasoning steps.

INT4 / INT8 4-bit and 8-bit integer quantization formats. Applying INT4 or INT8 quantization to KV cache entries before storing them in T1 or T2 can reduce transfer volume by 4–8×.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

JournalNode An InfraGraph node type introduced in this paper. Represents the ASJ journal storage pool with attributes: write throughput (IOPS), write latency (mean and P99), durability guarantee, and current occupancy. Edges to GPU nodes carry ASJ traffic class annotations for QoS priority assignment.

kagent A CNCF Sandbox project (accepted May 2025) that represents agents as Kubernetes custom resources, making per-agent session identifiers visible to the orchestrator and simplifying the mapping from agent sessions to fabric-level quotas.

KVFlow A distributed KV cache system that tracks each agent's steps-to-execution in an Agent Step Graph and prioritizes KV cache retention accordingly. Its workflow-aware approach directly informs PD-KVC's use of AgentTaskGraph as a scheduling input.

KV Cache (Key-Value Cache) Memory structure that stores attention key and value tensors computed during the prefill phase, eliminating redundant recomputation during autoregressive decoding. The central memory primitive of transformer inference. Ephemeral in classical serving; PD-KVC makes it persistent and fabric-managed.

LangGraph An agent framework from LangChain whose checkpoint mechanism is evidence of industry convergence on durable-execution patterns. A target runtime for ASJ fabric primitives.

LLM (Large Language Model) A transformer-based model trained on large text corpora to perform language tasks. The inference serving of LLMs is the workload context of this paper.

llm-d A Kubernetes-native distributed inference framework from Red Hat / IBM Research, referenced as a production system in the P/D disaggregation landscape.

LMCache An open-source distributed KV cache layer for enterprise LLM inference. One of several production systems whose design informs PD-KVC.

Mapping Decision Engine A component of the InfraGraph framework (defined in Section 7.3 of the AFPM whitepaper) that maps workloads to hardware resources. AgentTaskGraph nodes extend InfraGraph to give this engine the information needed to reason about agentic placement.

MCP (Model Context Protocol) Application-layer protocol for agent-to-tool communication. Published by Anthropic in November 2024, donated to the Linux Foundation's AAIF in December 2025. 97M+ monthly SDK downloads as of December 2025. Does not specify journaling semantics; ASJ operates below it.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

MemoryTierNode An InfraGraph node type introduced in this paper. Represents one of the four PD-KVC memory tiers with attributes: tier level (T0–T3), capacity, access latency, bandwidth, and eviction policy parameters.

MemServe A distributed KV cache system using an elastic memory pool. One of the production systems whose design informs PD-KVC.

MHA (Multi-Head Attention) The standard attention mechanism in transformers. KV cache size scales as $2 \times H \times L \times N \times d_{\text{head}} \times \text{dtype_bytes}$. At 128K tokens with large model configurations, KV cache exceeds the HBM of a single GPU.

MLA (Multi-head Latent Attention) An attention variant introduced in DeepSeek-V2 that compresses K and V into a low-rank latent representation, reducing KV cache size by up to 93% relative to MHA. Adopted in DeepSeek-V3, R1, and Kimi K2.

Mooncake A KV-cache-centric disaggregated inference architecture powering Moonshot AI's Kimi service. Introduced disaggregated prefill/decode in production and reports 75% more requests served versus its prior vLLM-based stack. The primary production reference for PD-KVC design.

NIC (Network Interface Card) The hardware component that connects a compute node to the fabric. In PD-KVC and ASJ, the NIC (or DPU) executes RDMA operations and, in the ASJ path, performs journal writes in parallel with tool call execution.

NIXL (NVIDIA Inference Transfer Library) NVIDIA's transfer library for KV cache movement in inference serving. Supports Mooncake as a plug-in backend.

NVIDIA Dynamo A Kubernetes-native distributed inference framework announced at GTC 2025. Provides disaggregated prefill/decode, LLM-aware request routing, multi-tier KV cache management with object store offload, and a dynamic Planner for rebalancing workers. A primary production reference for PD-KVC.

NVLink 6 NVIDIA's latest GPU-to-GPU interconnect, introduced with the Vera Rubin platform in 2026. A proprietary Scale-Up fabric alternative to UALink within NVIDIA GPU clusters.

NVMe (Non-Volatile Memory Express) A storage protocol for flash SSDs. PD-KVC T2 uses high-endurance NVMe SSDs accessed via GPUDirect Storage for persistent KV cache and HAC snapshots.

OCP (Open Compute Project) The foundation under which the AI HW/SW Co-Design Workgroup operates and this paper is published. Promotes vendor-neutral, open infrastructure standards.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

P/D Disaggregation (Prefill-Decode Disaggregation) A serving architecture that routes compute-bound prefill and memory-bandwidth-bound decode to separate GPU pools connected by high-speed fabrics. Productionized by Mooncake and NVIDIA Dynamo; integrated into SGLang, vLLM v1, and TensorRT-LLM via the Mooncake Transfer Engine.

PagedAttention vLLM’s mechanism for mapping the KV cache to non-contiguous physical memory pages, reducing internal fragmentation and enabling dynamic sequence length growth. Local to a single GPU node; provides no cross-session persistence or multi-agent sharing.

PCIe (Peripheral Component Interconnect Express) The standard host-device interconnect bus. PCIe 5.0 x16 provides ~64 GB/s per direction between GPU and CPU memory. This per-direction bandwidth is the binding constraint for KV cache offloading and makes PCIe offloading impractical at long context lengths.

PD-KVC (Persistent Distributed KV Cache) The vendor-neutral, agent-aware reference architecture proposed in this paper for KV cache management across a four-tier memory hierarchy (T0–T3). Key design principles: session-addressed namespace, RDMA-native data path, staging and prefetch ahead of each reasoning step, zero-copy multi-agent sharing, and session-aware eviction with reference counting.

Pipeline Parallelism A distributed inference strategy that splits model layers across multiple GPUs or nodes, passing activations between pipeline stages. Generates synchronous, latency-sensitive traffic; shares fabric links with PD-KVC T1 reads.

Planner Agent An agent that decomposes a task, assigns subtasks to specialized sub-agents (executor, verifier, critic), and coordinates their outputs. Requires persistent coordination state across the full multi-agent workflow.

PoD (Point of Delivery) A cluster grouping whose internal fabric (Scale-Up) provides lower latency and higher bandwidth than cross-PoD (Scale-Out) links. Placing an agent’s reasoning steps within the same PoD as its KV cache is a key PD-KVC optimization.

Prefill Phase The first phase of LLM inference: the model processes all input tokens in a single forward pass and computes KV vectors for each attention head at each layer. Compute-bound.

QoS (Quality of Service) Traffic management policies that assign different priority, bandwidth guarantees, and latency targets to different traffic classes. PD-KVC T1 reads require a new traffic class: latency-sensitive but small-message, distinct from both synchronous collective communication and asynchronous bulk transfers.

RAG (Retrieval-Augmented Generation) A technique that retrieves relevant context chunks from an external vector store using embedding similarity search, rather than attending over the

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

full context. Effective for knowledge retrieval but cannot represent ordered action histories or causal dependencies between agent reasoning steps.

RDMA (Remote Direct Memory Access) A network capability allowing a NIC or DPU to transfer data directly between remote and local memory without CPU involvement. Core transport for PD-KVC T1 operations and ASJ journal writes.

Restate A journal-based durable-execution engine whose journal-and-replay model directly informs ASJ semantics.

ROD / RUD (Reliable Ordered Delivery / Reliable Unordered Delivery) Two delivery modes in UEC 1.0. Proposed as the foundation on which the Durable Write primitive could be standardized as a new opcode class in the UET Semantic Sub-layer.

Runaway Agent An agent that enters an unproductive loop, consumes excessive resources, or takes unintended external actions due to model errors or adversarial inputs. Detected and contained by the Circuit Breaker mechanism.

Saga Pattern A distributed systems pattern in which a long-running transaction is decomposed into individually committed steps, each with a compensating action that can undo its effects if a later step fails. Referenced as a partial analogy for agentic recovery; difficult to apply when agent action sequences are open-ended and LLM-generated.

Scale-In Fabric The lowest-level interconnect domain in the 2026 OCP taxonomy. Typically intra-chip or intra-package (not the primary focus of this paper).

Scale-Across Fabric The highest-level interconnect domain in the 2026 OCP taxonomy, spanning data centers or wide-area links (not the primary focus of this paper).

Scale-Out Fabric The network layer connecting PoDs. Instantiated by UEC 1.0, which defines AI Full and AI Base profiles tuned to small-message AI traffic. Provides transport for PD-KVC T1 reads when agent sub-tasks span PoD boundaries. Higher latency than Scale-Up.

Scale-Up Fabric The network layer within a PoD. Instantiated by UALink (1.0 ratified April 2025; 2.0 ratified April 2026) and NVLink 6. Sub-microsecond latency with memory semantics. Primary target for PD-KVC T1 reads and writes.

Session Affinity The scheduling property of placing an agent's reasoning steps on GPU nodes within the same PoD as its cached KV state, minimizing fabric hop count and latency. Represented in InfraGraph via Session Affinity Annotations.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Session-Addressed Namespace PD-KVC's addressing scheme. Each agent task has a globally unique session identifier; KV entries are addressed by a compound key (session_id, layer_id, token_range). Immutable shared prefixes (e.g., a common system prompt) are content-hash addressed and deduplicated; mutable, agent-specific content is session-scoped.

SGLang An LLM inference framework that integrated the Mooncake Transfer Engine as a KV connector backend in 2025.

SSM (State-Space Model) A neural architecture alternative to transformers (e.g., Mamba) with fundamentally different state management. The PD-KVC tier hierarchy remains applicable but specific cache structures differ for SSMs.

Stateless Inference The classical LLM serving model: each request begins with a cold context, the model computes a completion, and the session ends. No state is preserved between requests. Agentic AI requires the opposite: persistent state that can be shared and retrieved efficiently.

T0 / T1 / T2 / T3 The four memory tiers of PD-KVC. T0: active KV cache in GPU HBM (~ns latency, TB/s bandwidth). T1: RDMA-accessible DRAM cluster pool (1–20 μ s). T2: NVMe SSD via GPUDirect Storage (10–200 μ s). T3: distributed object store for long-term episodic memory and audit logs (ms to s).

Temporal A durable-execution platform whose journal-event history and journal-based-replay model directly inform ASJ semantics. Integrated by LangGraph and Pydantic AI; referenced alongside AWS Lambda durable functions, Restate, DBOS, Cloudflare Workflows, and Inngest as established durable-execution platforms.

Tensor Parallelism A distributed inference strategy that splits individual weight matrices across multiple GPUs, requiring all-reduce communication at each layer. Synchronous and latency-sensitive; its all-reduce traffic shares scale-up links with PD-KVC T1 reads.

TensorRT-LLM NVIDIA's optimized LLM inference library. Integrated the Mooncake Transfer Engine as a KV connector backend in 2025.

TokenLake A distributed KV cache system using unified segment-level prefix addressing. One of the production systems whose addressing approach informs PD-KVC's session-addressed namespace.

Token The basic unit of text processed by an LLM. Context length, KV cache size, and inference cost are all measured in tokens.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Transformer The neural architecture underlying modern LLMs. Defines the attention mechanism that produces KV cache entries during inference.

Two-Phase Commit (2PC) A distributed protocol for atomic commitment across multiple participants. Proposed as a lightweight mechanism between ASJ and external tool backends for high-value, non-idempotent actions (e.g., financial transactions) where ASJ alone cannot guarantee exactly-once execution.

UALink® The Scale-Up interconnect standard for AI accelerator pods within a PoD. UALink 1.0 was ratified April 2025 (200G/lane, up to 1024 accelerators per pod). UALink 2.0 was ratified April 2026, adding In-Network Compute.

UEC / UET (Ultra Ethernet Consortium / Ultra Ethernet Transport) Industry consortium and transport standard optimized for AI traffic. UEC 1.0 defines AI Full and AI Base profiles for small-message AI workloads. The UET Semantic Sub-layer is the proposed location for a standardized Durable Write opcode class.

Vendor-Neutral Designed to work across hardware and software stacks from multiple vendors without dependency on any single vendor's implementation. A primary goal of PD-KVC, ASJ, and the InfraGraph extensions proposed in this paper.

Vera Rubin NVIDIA's GPU platform introduced in 2026, which includes NVLink 6 as its Scale-Up fabric and an Inference Context Memory Storage platform for rack-scale KV cache offload.

vLLM An open-source LLM serving library that introduced PagedAttention. The vLLM v1 release integrates the Mooncake Transfer Engine as a KV connector backend.

WAL (Write-Ahead Log) A database durability pattern in which changes are written to a sequential log before being applied to the main data store. ASJ borrows WAL semantics: the journal is the source of truth for the agent's causal history, and each AT is committed to the journal before the next action begins.

Zero-Copy Sharing A PD-KVC mechanism for multi-agent workflows. Sub-agents receive a reference to a shared session namespace rather than a physical copy of the KV cache entries, eliminating $O(N \times \text{num_agents})$ memory overhead for shared context prefixes. Reference counting governs eviction: shared blocks are eligible for eviction only after the last referencing session releases them.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Appendix B

Proposal: External Standardization Collaboration for the Agentic AI Fabric

OCP AI HW/SW Co-Design Workgroup

Companion to the 2026 whitepaper, AI Fabric Specialization for Agentic AI

Contributors: Weifeng Zhang, Victor Lu, Deep Shah

Purpose

This document lays out a set of collaboration tracks the workgroup can use to open talks with outside standards groups on the open items from the 2026 whitepaper, AI Fabric Specialization for Agentic AI. Treat it as a starting point for first outreach rather than a finished spec.

Why engage the external community now

The whitepaper's main proposals are PD-KVC, ASJ, and the InfraGraph extensions. All three are written to work across vendors rather than for one vendor alone, so they only pay off if many groups pick them up together. A PD-KVC design is worth little unless several inference stacks and fabric vendors can use it, and an ASJ Durable-Write step needs guarantees that reach across the NIC, the switch fabric, and the storage backend. Those areas already fall under the OCP, the Ultra Ethernet Consortium, the Linux Foundation, and the CXL Consortium, which is why we want to bring them in.

Timing also matters here. MCP and A2A, the agentic-AI protocols at the application layer, are still coming together under the Linux Foundation through 2025 and 2026. If we start while that work is in motion, our fabric-layer pieces can fit alongside it rather than end up repeating it later.

Proposed collaboration tracks

Each track ties an open item from the whitepaper to the outside group best placed to lead it, with a first ask to get the conversation going.

Track	Basis in whitepaper	External partner	Proposed initial ask
T1: Durable-Write fabric primitive	Section 4.3.B (ASJ)	Ultra Ethernet Consortium (UEC)	Work together to test a Durable-Write confirmation step in the UET Semantic Sub-layer,

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

			built on UEC 1.0 ROD/RUD delivery modes.
T2: Agent Transaction Schema	Section 4.4	Linux Foundation / Agentic AI Foundation (AAIF)	Propose a shared Agent Transaction format (intent and effect records) to go alongside MCP, which does not cover journaling.
T3: PD-KVC reference architecture	Section 3.3	OCP (internal first), then inference and fabric vendors	Bring the Mooncake, NVIDIA Dynamo, LMCache, and TokenLake versions together into one shared, vendor-neutral design that InfraGraph can model.
T4: Coherent memory pooling for the T1 tier	Section 3.4	CXL Consortium	Look at CXL 4.0 memory pooling as a way to back the T1 warm-pool tier where rack-scale memory sharing is available.
T5: InfraGraph agentic extensions	Section 7	OCP / InfraGraph maintainers	Agree on standard node and edge types for Memory Tier, Session Affinity, Agent Task Graph, ASJ Journal, and Circuit Breaker.

Workload environments and their effect on requirements

The tracks above assume one kind of setup, but the hardware and fabric needs behind each one shift with the environment a workload runs in. The same agentic building block can call for different guarantees depending on where and how it runs. Because of that, we think the work should cover a few environment types from the start, and we are putting this in the proposal rather than the whitepaper since it is about how needs change with the setup, not about the core design. Two environment types matter most here:

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Environment	How requirements change	Which tracks it affects
Kubernetes vs. non-Kubernetes	On Kubernetes, teams can use device plugins, schedulers, and operators to place and isolate workloads, and a growing set of CNCF and Linux Foundation projects already give agents and their traffic a native home there. On bare-metal or custom setups, the same work must be done at a lower layer.	Affects how PD-KVC placement (Track 3), InfraGraph session affinity (Track 5), and HAMi-style T0 isolation are handled. The same distinction shapes how network-fabric interworking (Track 6) and platform management (Track 8) get enforced, since device plugins and schedulers give Kubernetes a native enforcement point that bare-metal setups lack.
Data center vs. edge	Data-center nodes have plenty of HBM, fast fabrics, and rack-scale memory sharing. Edge nodes have tighter memory, power, and network limits, and often have no Scale-Up fabric at all.	Changes the bandwidth and latency targets for the PD-KVC tiers (Track 3, Track 4) and the durability path for ASJ journaling (Track 1). Changes the bandwidth and latency targets for the PD-KVC tiers (Track 3, Track 4) and the durability path for ASJ journaling (Track 1). Edge constraints also affect network-fabric interworking (Track 6), since the absence of a Scale-Up fabric changes which transport guarantees ACC can rely on.

Looking at each track through these two lenses early on should help the outside groups set hardware needs only where they are really needed, instead of treating one data-center, Kubernetes-based setup as the default for everything.

Open research questions to raise with the community

A few items from the whitepaper are still open enough that they make more sense as shared research questions than as finished proposals. Putting them on the table invites others to work on them with us:

- Making sure ASJ recovery behaves correctly, including what happens if a failure hits during the journal write itself.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

- Designing attention algorithms that work with paged, RDMA-based KV caches without giving up speed.
- Placement methods that hold up at clusters of thousands of devices.
- Congestion behavior of the high-frequency, small-message KV read class when it shares fabric links with all-reduce and bulk transfers, including incast avoidance and isolation, to be studied in the follow-up paper on this fabric.

Recommended sequencing

It makes sense to start with Track 1 (UEC) and Track 5 (InfraGraph). Both have a clear home and need little coordination, so we can show progress quickly. Track 6 (IETF/IRTF) can move alongside them, since a side meeting or non-WG-forming BOF is a similarly low-overhead way to start. Track 2 runs through the AAIF, which is still new, so it is better to pick that up once we have a contact there. Track 3 and Track 4 can move in parallel as vendor talks. Track 7 (TM Forum), Track 8 (DMTF), and Track 9 (CSA) are better suited to a second wave, once the workgroup has something concrete from Tracks 1, 3, and 5 to bring into those conversations. Whichever track we take first, it is worth checking it against the two environment types above, since the same building block may need different hardware at the edge than in the data center.

Acknowledgement

Special thanks to Mesut Oezdil for his great effort to draft this proposal and various other contributions to this whitepaper.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Compliance with Open Compute Project Tenets

Openness

This paper addresses new AI fabric design requirements due to agentic AI behaviors, which are not addressed in the 2026 OCP “AI Fabrics Perspective and Modeling” whitepaper. It identifies gaps and extensions to the existing AI fabric framework that represent open research directions for the OCP AI HW/SW Co-Design Workgroup and the broader community.

Efficiency

This paper makes proposals on AI fabric extension to support long-context and stateful memory management, and fault tolerance & recovery of stateful agent tasks.

Impact

This paper identifies the research directions specifically for agentic AI fabric, which sit directly at the intersection of agentic AI runtime behavior and AI fabric design. It proposes extensions to the existing AI fabric and communication protocols to support new agentic behaviors.

Scale

The extensions proposed in this paper enable agents to efficiently execute across multiple reasoning steps and operate over extended time horizons.

Sustainability

This contribution supports sustainability of AI infrastructure through reliable agent execution via fault tolerance and recovery for stateful agent tasks.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

About Open Compute Project

The Open Compute Project (OCP) brings at-scale innovations and hyperscaler best practices to all, spanning technology domains from the data center to the edge, and the technology stack from silicon, to systems, to site facilities and services. The international OCP Community is made up of organizations and people from hyperscale and tier-2 cloud data center operators, communications providers, colocation providers, diverse enterprises, and technology vendors. The OCP Foundation fosters collaboration within the Community to tackle market challenges in an array of OCP Projects that create open contributions such as specifications, designs, and more. The OCP Solution Provider Program then recognizes the application of those contributions in compliant products, solutions, and data center facilities and services with designations like OCP Inspired, OCP Accepted and OCP Ready. With the tenets of openness, impact, efficiency, scale and sustainability, the OCP engages and educates thousands of engineers every year through many events and webinars. Across many initiatives the OCP Foundation and Community are meeting the market today and shaping the future.

Learn more at: www.opencompute.org.

Date: September 2026



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).