Vectorized Communication in Distributed LLM Systems: Scenarios, Challenges, and Design Space

Jiaxun Lu, Yunfeng Shao, Xu Wei, Yin Jun

Abstract—Distributed LLM systems increasingly exchange intermediate representations across compute, storage, and network boundaries. This survey develops a communication-centered view of these transfers, covering parameters, activations, KV caches, hidden states, and latent messages across distributed training, disaggregated inference, cache pooling, and multi-agent interaction. We organize the literature along four usage perspectives and three fidelity tiers: exact-state reconstruction, lossy compression, and semantic collaboration. The survey relates these settings through task utility, transferred bytes, end-to-end latency, and redundant recomputation. To make this trade-off explicit, we introduce Semantic Density (SD), a task-centered reporting protocol that measures retained utility per normalized transferred byte relative to an uncompressed exact-state reference. Using this framework, we identify recurring challenges in wide-area communication, state reuse, latent-space alignment, and transport–memory coordination, and organize existing approaches around compression, reuse, semantic communication, and storage-aware transfer. We conclude by outlining open problems in alignment, security, fidelity evaluation, standardization, economics, and communication beyond Transformers.

Index Terms—Distributed AI systems, vectorized data transfer, compute–network trade-off, semantic fidelity, multi-agent coordination

1 Introduction

LLM inference has moved beyond monolithic serving toward increasingly fine-grained disaggregation. Prefill–decode (PD) separation has evolved from a proposed optimization into a deployed serving pattern. Prefill processes the prompt and produces the KV cache, whereas decode generates tokens autoregressively and repeatedly reads that cache. Separating the two phases enables independent resource allocation and makes KV-cache transfer a first-class systems path [1], [2], [3].

At larger scales, Mooncake demonstrates KV-centric disaggregated serving with pooled CPU, DRAM, and SSD resources, while Cross-Datacenter Prefill–Decode Disaggregation (PDD) extends PD across data centers and combines heterogeneous scheduling with relay and recomputation to manage cross-domain transfer [3], [4]. The granularity of disaggregation is also increasing. For MoE models, MegaScale-Infer separates attention and FFN or expert execution so that the two parts can use different parallelism and resource pools [5]. Recent Attention–FFN Disaggregation (AFD) work further studies when this finer split pays off under workload, SLO, and network constraints [6]. In parallel, multi-agent systems such as AutoGen [7] and ReAct [8] distribute complex tasks among specialized agents, creating additional demand for sharing intermediate state.

As models and agents span nodes and domains, inference must move intermediate representations such as KV caches, hidden states, embeddings, and latent messages. PagedAttention already treats the KV cache as an explicit memory-management object [9]. More broadly, hardware heterogeneity, energy limits, data locality, and governance requirements may place different stages in different resource pools or jurisdictions [1], [10]. The resulting systems path crosses compute, storage, and network layers, making transfer cost and representation compatibility central design concerns.

Text-based interaction requires the sender to decode internal computation into tokens and the receiver to encode those tokens and repeat prefill. Direct vector transfer can preserve or reuse more intermediate state, but it introduces transfer overhead, alignment requirements, and security concerns. This trade-off is related to the semantic-communication principle of transmitting task-relevant meaning rather than raw bits [11]. Existing surveys typically focus on one layer, such as KV-cache management, end-to-end inference, or agent communication. They do not place model-state transfer, storage, network cost, and task-level fidelity in a single organizing view [12], [13], [14], [15], [16], [17].

1.1 Vectorized Communication: An Organizing Perspective

The vectorized communication discussed in this survey is not a specific protocol. It is a systems pattern in which intermediate representations cross compute, storage, or network boundaries. We use four descriptive tags to organize where these representations originate, how they are stored, and how they support interaction.

As a running example, consider a multi-agent retrieval-augmented generation request in which a planner and a

Corresponding author: Yunfeng Shao. Jiaxun Lu and Yunfeng Shao are with Huawei, Shenzhen, China (e-mails: lujiaxun@huawei.com, shaoyunfeng@huawei.com). Xu Wei and Yin Jun are with COSCO Shipping, China (e-mails: xu.wei3@coscoshipping.com, yin.jun1@coscoshipping.com).

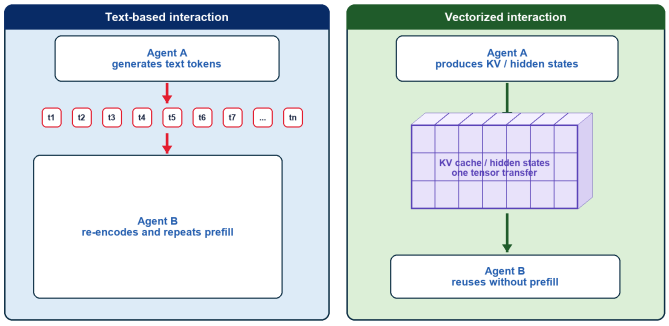

Fig. 1. Illustrative multi-agent interaction over shared context. Text-based exchange may require re-encoding and repeated prefill, whereas compatible vectorized exchange can reuse an intermediate state at the cost of additional communication.

verifier access overlapping long-context evidence. Figure 1 contrasts text-based and vectorized interaction in this setting.

Vectorized intelligence. Transformers tokenize multimodal inputs (text, images, audio) into high-dimensional vectors and perform information mixing and feature extraction in vector space through Query/Key/Value self-attention [18]. Intelligent tasks are computations over vector space, and vectorization is the substrate on which intelligence arises.

Vectorized memory. In PD-disaggregated and multi-agent settings, the KV cache is the explicit cache of attention history and serves as the physical carrier of *memory* [9]. Transmitting KV caches instead of repeating prefill means that memory flows between nodes in tensor form, which is the most direct manifestation of vectorized memory [2], [3].

Vectorized thinking. Latent reasoning replaces some discrete reasoning tokens with one or more continuous latent states. In selected tasks, this can reduce the number of generated reasoning steps, although the payload size and model compatibility remain important constraints. Representative work includes Coconut, SoftCoT, and CODI [19], [20], [21]. See Section 6.3.

Vectorized collaboration. When multi-agent systems communicate through natural language, the receiver may repeat encoding and prefill computation. When compatible agents exchange KV caches or latent states directly, part of the sender's intermediate computation can be reused, trading communication and alignment overhead for computation [22], [23], [24], [25], [26].

These four tags expose a recurring trade-off. Vectorization can reduce computation while increasing communication. Latent reasoning reduces reasoning tokens, and KV reuse removes repeated prefill, but their intermediate states may be larger than the original token sequence. The additional bytes can carry more task-relevant information, but whether they improve task accuracy depends on the model and workload (see Section 6.3). Classical communication targets bit-level losslessness [27]. Vectorized transfer can instead target task-sufficient semantic fidelity [11]. The systems problem is therefore to choose an operating point between compute cost and communication cost.

Accordingly, the survey traces three linked questions. Which cross-domain communication scenarios arise, and what do they require? Which technical challenges shape compression, reuse, semantic communication, and storage? Which obstacles still separate research prototypes from production systems?

Our contributions are as follows. (1) We use vectorized communication as an organizing perspective, characterize interaction with four descriptive tags, and classify communication modes by distortion tolerance. (2) We formulate Semantic Density as a task-centered reporting protocol and use it with end-to-end latency and redundant recomputation to characterize the transfer Pareto frontier. (3) We compare transfer requirements across training, inference, pooling, and multi-agent interaction. (4) We identify four technical challenges: bandwidth limits, reuse beyond exact prefixes, latent-space alignment, and cross-layer coupling. (5) We map the design space through four optimization routes: compression, reuse and caching, latent or semantic communication, and hierarchical storage with prefetching. (6) We discuss seven open directions spanning alignment, security, fidelity, standardization, economics, transferability, and post-Transformer architectures.

The remainder of the paper is organized as follows. Section 2 introduces basic concepts and presents the literature landscape. Section 3 establishes the scenario taxonomy. Section 4 defines the Semantic Density reporting protocol, the transfer Pareto frontier, and scenario-specific requirements. Section 5 identifies the constraints that limit this frontier. Section 6 maps the design space through optimization routes that advance it. Section 7 discusses open problems and future directions. Section 8 synthesizes the main findings and open directions.

2 Preliminaries and Literature Landscape

2.1 Basic Concepts

KV cache. A KV cache stores the Key and Value vectors of historical tokens in a Transformer decoder. It avoids recomputation during decoding and is a central data structure for inference performance and memory use [9].

Prefill phase. The prefill phase processes the input prompt and produces the KV cache needed for the first output token. It is compute-intensive [1], [2].

Decode phase. The decode phase generates subsequent tokens autoregressively and repeatedly reads the KV cache. It is memory-bandwidth-intensive [1], [2].

Prefill-Decode disaggregation (PD disaggregation). PD disaggregation deploys the prefill and decode phases on different compute nodes to enable phase-independent resource allocation and scaling [2], [3].

Latent communication. Agents directly exchange continuous model-state representations (embeddings, hidden states, or KV caches) rather than communicating through discrete natural-language tokens.

We use D_F for the byte volume of a transferred representation F, T_{comp} for local recomputation time, B_{eff} for effective wide-area bandwidth, and T_{overhead} for encoding, queueing, memory-copy, and retransmission overhead. RTT denotes round-trip time. For method m, P_m is downstream task utility, B_m is transmitted cross-domain bytes, L_m is end-to-end latency, and C_m is redundant local recomputation cost. SD denotes Semantic Density, defined in Section 4.1.

2.2 Literature Landscape

We organize the surveyed literature into six branches (Figure 2). Four branches correspond to descriptive usage tags, and two supporting branches cover cross-domain scenarios and infrastructure, as well as theoretical foundations and future directions. The tree covers distributed training, disaggregated inference, KV-cache pooling, agent memory, latent reasoning and message passing, cache reuse and compression, hierarchical storage, semantic communication, and architectures beyond the Transformer.

2.3 Survey Scope and Selection

To contextualize the rapid growth of research on distributed LLM training and inference with cross-device communication, we first performed a broad retrieval on arXiv in August 2026 using the query `LLM` combined with any of `KV cache`, `disaggregated inference`, `prefill decode`, `latent communication`, `semantic communication`, or `multi-agent`. Grouped by submission year, this query returned 96, 691, 2,180, and 2,468 candidate records for 2023 through 2026, respectively (2026 counts cover submissions through August). As shown in Figure 3, the candidate volume has increased by more than an order of magnitude over this short period, reflecting surging interest in communication-centric challenges beyond single-node or single-datacenter settings.

From this body of work, we conducted a focused survey. We searched arXiv, DBLP, ACM and IEEE digital libraries, OpenReview, and major conference proceedings using the same keyword combinations, followed by backward and forward citation tracing. We retained studies on communication mechanisms, systems optimizations, or evaluations involving parameters, activations, KV caches, hidden states, or latent messages across devices, clusters, or regions. We excluded work focused exclusively on isolated model compression or single-node kernel optimization. The resulting snapshot through August 2026 comprises 86 cited works.

The four usage branches identify where representations are generated and used. The cross-domain branch groups them by fidelity requirement and records reuse and storage mechanisms. The theory-and-future branch covers information theory, semantic communication, computing-power networks, regional scheduling, and alternative generation architectures. The detailed mappings appear in the scenario and optimization tables below.

Table 1 positions this survey relative to representative reviews. Existing work covers KV-cache management, general inference systems, agent memory, and latent communication, whereas our focus is the communication-centered interaction among vectorized state, network, storage, and task semantics.

3 Vectorized Communication Scenarios and Taxonomy

3.1 Vectorized Interaction Scenarios

Cross-wide-area vectorized interaction is not confined to PD-disaggregated inference. It spans training and inference, single models, and multi-agent systems. Four scenario families can be identified by interaction carrier.

(1) Vectorized interaction in distributed training. Cross-wide-area training differs from intra-cluster parallelism because synchronization must tolerate limited bandwidth, non-negligible RTT, heterogeneous workers, and intermittent failures. Data-parallel training communicates model updates, while pipeline-parallel training communicates activations and gradients. DiLoCo reduces the frequency of parameter synchronization by performing multiple local optimization steps between global updates [28]. DiPaCo co-designs modular path composition with DiLoCo to support poorly connected and heterogeneous workers [29]. OpenDiLoCo provides a globally distributed implementation and demonstrates training across multiple continents and countries [30]. Photon applies cross-silo federated training to global-scale LLM pretraining and reduces communication relative to conventional distributed baselines [31]. Decoupled DiLoCo relaxes lock-step synchronization through asynchronous learner updates and quorum-based aggregation [32]. Agora extends internet-scale training toward bandwidth-efficient pipeline-parallel model sharding and fault-tolerant collective operations over internet-grade links [33]. These systems make parameters, optimizer updates, and gradients the primary vectorized communication objects, with convergence, goodput, and communication volume as the central requirements. DiLoCo scaling studies further examine how the low-communication approach changes with model size and compute budget [34].

(2) Cross-wide-area distributed inference. This family contains three related inference architectures. Model-split inference partitions the model by layer or module across cloud, edge, and terminal devices, and transfers activations between stages. AI Flow, CROSS-SEC, and CE-LSLM are classified as exact-state activation transfer [35], [36], [37].

Operator-level systems use a finer split. They transfer hidden activations between separately provisioned attention and FFN or expert pools. MegaScale-Infer and AFD expose the same exact-state communication problem within MoE serving [5], [6].

PD-disaggregated inference separates prefill and decode across clusters. The transferred object is the KV cache rather than an intermediate activation. Mooncake provides a KV-cache-centric architecture [3]. P/D-Serve studies large-scale disaggregated serving with device-to-device KV transfer [38]. PDD uses a relay architecture [4]. KVDirect focuses on tensor-centric transfer [39]. PrfaaS abstracts prefill as a schedulable cross-data-center service [40]. FlowKV combines low-latency KV transfer with load-aware scheduling [41].

(3) KV cache pooling and agent memory pools. Treating the KV cache as a poolable storage resource is the systematic embodiment of *storage for compute*. Mooncake builds a hierarchical KV pool over CPU, DRAM, and SSD [3]. Alibaba Cloud's Tair KVCache elevates the KV cache to an enterprise-grade global cache service. It implements *storage in place of compute* through HBM/DRAM pooling and tiered caching [42]. Extended to wide-area settings, such pooled storage can evolve into cross-region KV storage and transfer services, enlarging the freedom of cache reuse and scheduling.

On the multi-agent side, vectorized storage of explicit

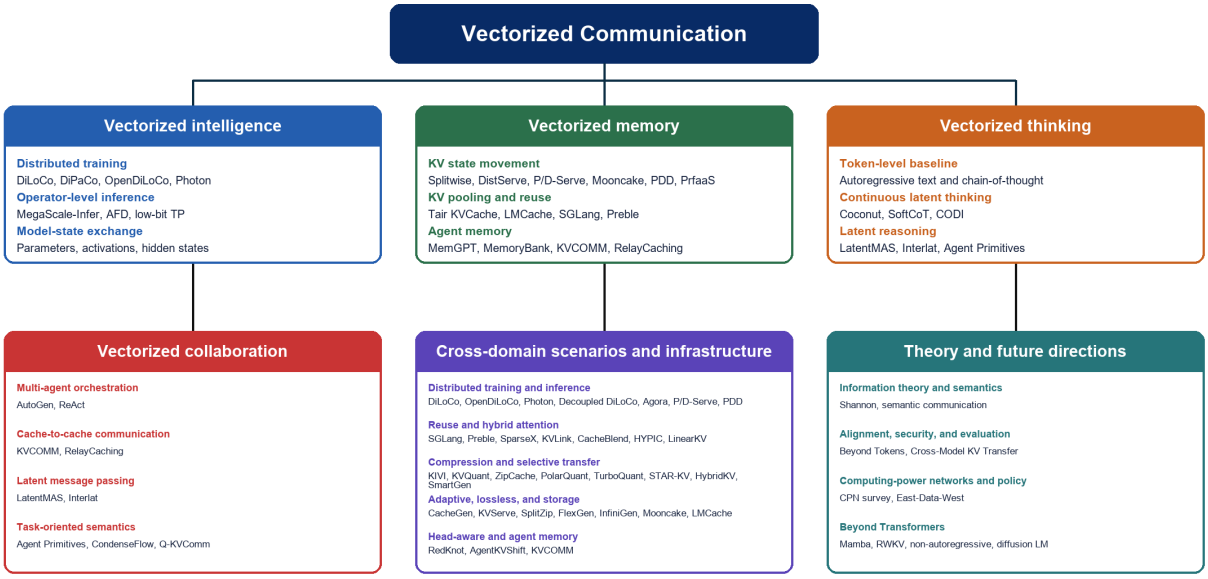

Fig. 2. Literature landscape organized into four descriptive usage perspectives and two supporting branches for cross-domain infrastructure and future directions.

TABLE 1
Positioning relative to representative surveys.

Survey	Primary scope	Communication object	Distinct emphasis
KV-cache surveys [12], [13]	KV management and compression	KV tensors	Memory reduction, fidelity, and serving efficiency
LLM inference-system surveys [14]	End-to-end inference systems	Requests, activations, and KV caches	Operators, scheduling, memory, and system composition
Agent-memory and communication surveys [15], [16], [17]	Agent memory and communication protocols	Memory states, messages, and latent representations	Memory mechanisms, message objects, alignment, and collaboration
This survey	Cross-domain vectorized communication	Parameters, activations, KV caches, and latent states	Scenario taxonomy, network–compute trade-offs, and task-centered evaluation

memory is emerging. MemGPT treats the LLM as an operating system and manages external memory hierarchically [43]. MemoryBank updates long-term memory over time [44]. A recent survey reviews the mechanisms and evaluation of agent memory [15]. Vectorized memory can reduce information loss relative to text-based memory and can avoid repeating prefill over historical content, trading storage and communication for computation. Multi-agent systems also support direct cache-to-cache communication. KVCOMM reuses KV caches over shared context across agents [22]. RelayCaching reuses the KV cache produced in one agent's decode phase for the next agent's prefill phase [23].

(4) Multi-agent latent-variable collaboration. Agents interact through latent variables, such as latent states or KV caches, rather than text. This family covers latent thinking with Coconut [19], SoftCoT [20], and CODI [21]. See Section 6.3. It also covers latent message passing with LatentMAS [24] and Interlat, as well as vectorized interaction patterns with Agent Primitives [26]. Knowledge distillation can also be viewed as a form of vectorized knowledge transfer [45]. See Section 6.3.

Across these scenarios, vectorization replaces some repeated computation with communication and storage. In the running retrieval-augmented generation example, the planner and verifier may reuse a shared KV state, transfer a compressed state, or exchange a task-level latent message, depending on the required fidelity and the available bandwidth. The resulting trade-off, rather than a universal preference for transfer, is the common design question.

3.2 Classification Perspective and Taxonomy

Section 1.1 introduces four descriptive perspectives. They are intelligence, memory, thinking, and collaboration. These perspectives characterize the functional role of a representation, but do not specify the requirement for transferring it. Within any perspective, a receiver may need an exact state, an approximate state, or only task-relevant information. We therefore add required reconstruction fidelity as a second classification axis, with three tiers from strict to relaxed. These tiers are exact-state transfer, lossy compression, and semantic collaboration.

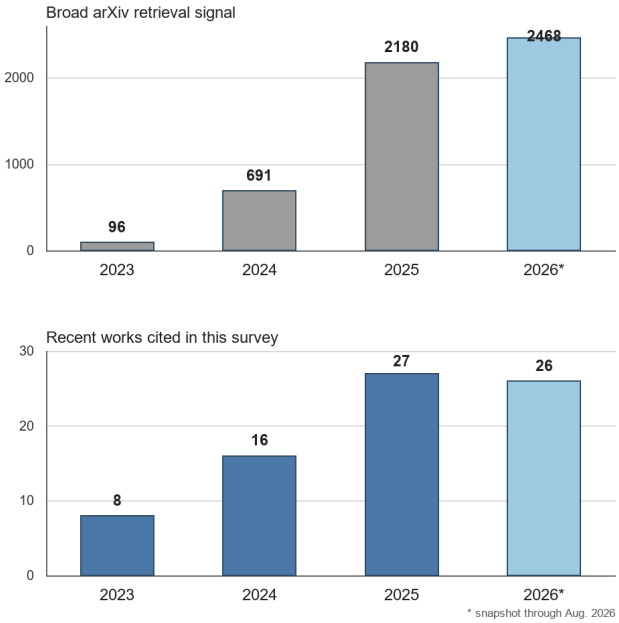

Fig. 3. The broad arXiv retrieval signal and the recent works cited in this survey both concentrate in recent years. The query returns 96, 691, 2,180, and 2,468 candidate records in 2023–2026, while the survey cites 8, 16, 27, and 26 works from those years. The remaining 9 cited works are earlier foundational references. The 2026 counts cover submissions through August 2026.

The two axes answer different questions. The descriptive perspective indicates where and why a representation is used, whereas the fidelity tier specifies how closely the receiver must recover it. They are complementary rather than strictly orthogonal. Hybrid systems may span multiple perspectives, but each table entry is placed by its dominant perspective and required fidelity.

Table 2 summarizes representative systems in this two-dimensional space. A dash indicates that we found no representative work under the primary scope of this survey. The three fidelity tiers are then discussed in detail in the following subsections.

3.3 Exact-State Transfer

Exact-state reconstruction requires the receiver to obtain a bit-identical or near-lossless KV cache tensor, is sensitive to packet loss and quantization noise, and applies to scenarios in which the attention state must be reconstructed precisely. Typical sub-scenarios include the following.

(1) KV transfer in PD disaggregation. The KV cache produced by the prefill node must reach the decode node before the first output token is generated [2], [3], [39], [41]. In wide-area deployments, the prefill node may sit at the edge and the decode node in the central cloud. For a model with sequence length L, N_{layer} layers, $N_{\text{KV-head}}$ key–value heads, head dimension d_{head}, and b bytes per element, the KV volume is

$$D_{\text{KV}} = 2LN_{\text{layer}}N_{\text{KV-head}}d_{\text{head}}b, \tag{1}$$

where the factor 2 accounts for Key and Value. Under MHA, $N_{\text{KV-head}}d_{\text{head}}$ equals the hidden size. GQA and MQA use fewer KV heads and reduce the per-token volume. DeepSeek's Multi-Head Latent Attention (MLA) compresses the Key and Value states into a lower-dimensional latent vector, reducing the cache further [62]. DeepSeek-V4 extends this direction with hybrid Compressed Sparse Attention (CSA) and Heavily Compressed Attention (HCA). At a one-million-token context, it reports 10% of the KV-cache footprint of DeepSeek-V3.2 [63]. These designs reduce the per-token payload but do not remove its dependence on sequence length. Even a highly compressed cache can therefore remain a substantial cross-domain payload for long-context requests.

(2) KV cache sharing across agents. Multiple agents that process overlapping context, such as retrieval-augmented generation over shared documents or multi-turn dialogue, can reuse each other's KV cache. KVCOMM targets such workloads with an online cross-context KV communication framework that reuses overlapping caches and calibrates cross-agent cache offsets [22]. In a five-agent setting, it achieves a cache reuse rate above 70% and up to 7.8× speedup. TTFT decreases from about 430 ms to about 55 ms [22].

(3) KV cache migration for cross-region load balancing. To optimize TTFT, cross-region schedulers need to route requests carrying warm caches to the corresponding nodes, or migrate caches along with requests. Preble proposes distributed prompt scheduling for long-prompt sharing, reducing average latency by 1.5–14.5× over existing serving systems through cross-node shared prefix caches [47].

(4) Model-split activation transfer. Model-split inference is also exact-state communication. The receiver continues the sender's forward computation and therefore requires functional compatibility. AI Flow, CROSS-SEC, and CE-LSLM partition the model across cloud, edge, or terminal devices and transfer activations between stages [35], [36], [37]. We classify these systems as exact-state activation transfer even when the application goal is semantic collaboration.

For PD disaggregation and KV-sharing cases, the transferred object is the Key and Value matrices. Model-split systems instead transfer activations between partitioned stages. Practical systems may further transfer only selected, compressed, or reused blocks.

3.4 Lossy Compression

Lossy compression targets bandwidth-constrained scenarios and exploits statistical redundancy in KV caches or hidden states (low-rank structure, outlier distributions) by quantizing, dropping, or decomposing at the feature level. The receiver does not require bitwise recovery, only a representation sufficient to generate coherent text. Representative technical routes include the following.

(1) Quantization. KV caches are quantized from BF16 to 4-bit, 2-bit, or lower. KIVI proposes tuning-free asymmetric 2-bit quantization. It quantizes Key caches per channel and Value caches per token, reducing peak memory while preserving generation quality [49]. KVQuant designs low-bit quantization schemes for outlier distributions to support extremely long contexts [64]. ZipCache achieves mixed-precision quantization through salient-token identification and channel-separable quantization [65]. TurboQuant stud-

TABLE 2
Scenario matrix using the dominant descriptive perspective (columns, Section 1.1) and primary fidelity tier (rows, Section 3.2). Hybrid systems may have secondary perspectives. A dash indicates sparse coverage under the primary scope of this survey.

Fidelity tier	Intelligence	Memory and reuse	Thinking	Collaboration
Exact-state	Distributed training and disaggregated inference: DiLoCo, DiPaCo, OpenDiLoCo, Photon, Decoupled DiLoCo, Agora, AI Flow, MegaScale-Infer, AFD, DistServe, P/D-Serve, PDD, KVDirect, FlowKV [2], [4], [5], [6], [28], [29], [30], [31], [32], [33], [35], [38], [39], [41]	KV-state storage and reuse: Mooncake, Tair KVCache, LMCache, Preble [3], [42], [46], [47]	Token-level autoregressive reasoning: text or CoT states, used here as an exact-state baseline [18], [19]	Cache-to-cache: KVCOMM, Relay-Caching [22], [23]
Lossy	Low-bit activation and tensor communication: tensor-parallel feature quantization [48]	KV compression: KIVI, TurboQuant, STAR-KV, HybridKV, KVServe, SmartGen, CacheGen, SplitZip [49], [50], [51], [52], [53], [54], [55], [56]	—	Adaptive KV compression: Q-KVComm [57]
Semantic	Semantic communication, latent features, and N-gram embeddings [11], [16], [58], [59], [60]	Fixed-length semantic KV: CondenseFlow [61]	Latent-space thinking: Coconut, SoftCoT, CODI [19], [20], [21], knowledge distillation [45]	Latent message passing: LatentMAS, Interlat, Agent Primitives [24], [25], [26], Beyond Tokens [16]

ies online vector quantization with near-optimal distortion and applies it to low-bit KV-cache quantization [50].

(2) Low-rank decomposition. KV caches are decomposed into low-rank representations. STAR-KV proposes a differentiable soft-thresholding mechanism for adaptive rank selection at the attention-head and block levels. It combines this mechanism with data statistics for near-lossless low-bit quantization. The method compresses the KV cache to 25% of its original size, up to 20× when combined with quantization. It also accelerates attention computation by up to 6.9× [51].

(3) Selective transfer. Only the KV entries that matter for generation quality are transferred. SmartGen partitions KV caches into universally important, context-dependent, and secondary categories, and transfers them via offline pre-push, parallel pull, and idle-bandwidth paths respectively, reducing TTFT by up to 4.3× relative to full KV transfer while maintaining near-lossless generation quality [54].

(4) Streaming compression. Compression is applied adaptively during online generation and transfer. CacheGen encodes KV caches into compact bitstreams and transmits them with adaptive chunked streaming, achieving 3.5–4.3× bandwidth savings while preserving near-lossless generation quality [55].

(5) Service-aware adaptation. Rather than applying a fixed compression profile, the system selects a profile according to current bandwidth, SLO, and quality constraints. KVServe formulates this choice as a service-aware control problem and adapts the compression level to changing serving conditions [53].

3.5 Semantic Collaboration

High-level semantic collaboration targets multi-agent collaborative reasoning. The communicated content leaves the per-token key-value level and is intended to support a task rather than to reconstruct the sender's internal state exactly. We use *semantic* here to describe the communication objective, not to claim that a latent vector is directly interpretable or model-agnostic. The receiver still requires a compatible decoder, fusion rule, or alignment mechanism. Typical subscenarios include the following.

(1) Latent thinking transfer. Agents reason over last-layer hidden embeddings rather than generating text and passing it. Interlat passes the last hidden state of the LLM directly as a continuous representation of *thought*. It reports up to 24× inference speedup on long-context multi-agent tasks [25]. LatentMAS proposes a training-free end-to-end latent collaboration framework that maintains and passes internal representations through a shared latent working memory. It reports up to 14.6% accuracy improvement across nine benchmarks, a 70.8%–83.7% reduction in output token usage, and end-to-end inference speedups of 4–4.3× [24].

(2) Semantic compression of KV caches. Lightweight modules compress KV caches into fixed-size semantic representations. CondenseFlow introduces a Latent Thought Condenser that compresses KV caches into fixed-length representations independent of context length via learnable semantic probes. Across seven benchmarks, it reduces KV cache memory by over 99%. It also reduces inference latency by about 20% relative to dense transfer and outperforms text-based methods by an average of 1.7 points [61].

(3) KV cache interaction. Agents pass reasoning context directly through KV caches. Agent Primitives abstracts multi-agent architectures into three reusable latent computation patterns (Review, Voting & Selection, Planning & Execution) and passes review context, option comparison, and planning intent through KV caches, improving accuracy by 12.0–16.5 percentage points over a single-agent baseline [26].

4 Semantic Density, Pareto Trade-offs and Transfer Requirements

Vectorized communication seeks a Pareto balance among task-relevant information, end-to-end latency, and redundant recomputation. We define Semantic Density as the

task utility retained per transmitted byte. End-to-end latency captures the service-level cost of moving and using the representation. Redundant recomputation captures the compute cost avoided by state transfer, reuse, or semantic abstraction. Accordingly, the requirements in this section are organized around four quantities. They preserve task utility P_m, limit transferred bytes B_m, control end-to-end latency L_m, and reduce redundant recomputation C_m. SD provides the utility-per-byte axis, while L_m and C_m capture system costs that SD alone does not capture. This section defines these quantities, examines their trade-offs, and then derives scenario-specific transfer requirements.

4.1 Semantic Density

Conventional metrics such as compression ratio, bandwidth use, and perplexity are incomplete for cross-wide-area scenarios. They measure transfer volume or representation fidelity, but not whether communication improves the downstream task. Existing KV cache compression work often uses perplexity or generation quality as a fidelity proxy [49], [51]. Task-oriented settings such as multi-agent collaboration also require task-level evaluation.

We formulate Semantic Density (SD) as a reporting protocol rather than a universal benchmark metric. For method m, let P_m denote downstream task utility and let B_m denote the total application-level bytes that cross the device, cluster, or regional boundary. The task metric underlying P_m is scenario-dependent, such as accuracy, exact match, task success rate, reward, or a generation-quality score. Let P_{ref} denote the task utility of a fixed high-fidelity reference execution under the same model, task, and input, and let B_{ref} denote the byte volume of the corresponding uncompressed exact-state transfer. We define $\widehat{P}_m = P_m/P_{\mathrm{ref}}$ and $\widehat{B}_m = B_m/B_{\mathrm{ref}}$, and report

$$\mathrm{SD}_m = \frac{\widehat{P}_m}{\widehat{B}_m}. \tag{2}$$

For the uncompressed exact-state reference execution, $\mathrm{SD}_{\mathrm{ref}} = 1$. We assume $P_{\mathrm{ref}} > 0$. If the task metric is a reward that can be zero or negative, the evaluation should use a positive utility transformation fixed before comparing methods. Intuitively, SD measures the task-relevant information conveyed per transmitted byte relative to this reference. Retained task utility is the operational proxy for this information. Here, $\widehat{P}_m$ is the fraction of reference task utility retained, whereas $\widehat{B}_m$ is the fraction of reference transfer volume used. A higher SD therefore indicates that a method retains task utility with fewer transferred bytes relative to the fixed reference. SD can also increase through representation enrichment. That is, if a method raises task utility while keeping boundary bytes fixed, P_m increases without a corresponding increase in B_m.

From an information-theoretic perspective, this protocol reframes *transfer fidelity*. Classical communication theory targets bit-level losslessness [27], with a distortion measure independent of the downstream task. Semantic communication instead emphasizes semantic fidelity [11]. For vectorized transfer, quantization, compression, and selective dropping should therefore be judged by whether they preserve task-relevant semantics, not only by whether they preserve every bit. SD expresses this perspective by relating task utility to communication volume and by replacing a universal *information-lossless* constraint with a task-dependent semantic-fidelity requirement.

Under the SD view, the three tiers may have different qualitative profiles. Exact-state transfer preserves the most task utility and therefore tends to keep P_m close to the reference, but may move task-irrelevant attention details and incur a large B_m. Moderate lossy compression can reduce B_m while retaining task quality, whereas excessive compression can reduce P_m and cause semantic collapse. Semantic collaboration may exchange task-level information more directly and improve the utility-per-byte ratio, but its P_m depends on cross-model alignment and fusion.

SD should be utilized jointly with end-to-end latency L_m and redundant recomputation cost C_m. A higher SD does not by itself guarantee lower latency or less recomputation. Evaluations should therefore report SD alongside the raw values of P_m, B_m, L_m, and C_m. SD comparisons should use the same model, task, input distribution, communication boundary, reference configuration, utility metric, and normalization.

Building on the SD perspective, the following two subsections examine whether a high-SD representation also yields an efficient system design. The first complements SD with the latency and recomputation axes by comparing transfer against local reconstruction. The second examines how representation choices change task utility and communication volume.

4.2 Transfer versus Full-Prefill Recomputation

In a cross-wide-area deployment, a receiver can either transfer an existing representation or reconstruct it locally. This choice is not specific to KV caches. It applies to all vectorized workloads. Let F denote the vectorized representation to be transferred, such as a KV cache, hidden state, or latent variable. Let D_F denote its data volume. Let T_{comp} denote the time for the receiver to recompute this representation. Let B_{eff} denote the effective wide-area bandwidth. RTT denotes the network round-trip time. Finally, T_{overhead} denotes encoding, decoding, queueing, memory-copy, and retransmission overhead. Under a single-flow, no-overlap approximation, the transfer time is

$$T_{\mathrm{transfer}} = \frac{D_F}{B_{\mathrm{eff}}} + \mathrm{RTT} + T_{\mathrm{overhead}}. \tag{3}$$

For full prefill recomputation, the attention component grows quadratically with sequence length. We use the first-order approximation

$$T_{\mathrm{comp}}(L) \approx aL + bL^2, \tag{4}$$

where the linear term captures projections and other per-token work, and the quadratic term captures full-sequence self-attention [66]. The stored KV payload grows approximately linearly with L. This difference is the reason transfer can become preferable for long contexts even after KV-cache reduction.

When $T_{\mathrm{transfer}} < T_{\mathrm{comp}}$, transfer is preferred under this cost model. This comparison assumes a single request and no overlap between communication and computation.

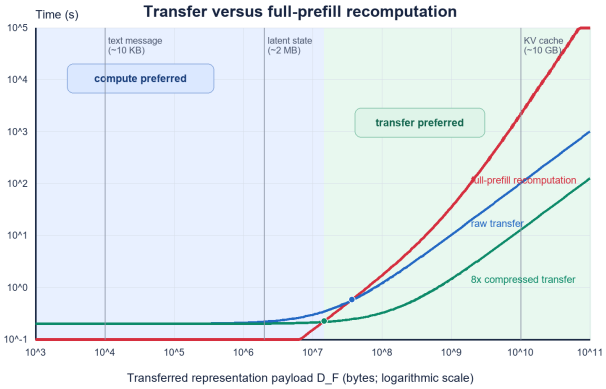

Fig. 4. Illustrative transfer-versus-full-prefill-recomputation trade-off under the cost model. The transfer curves follow Eq. 3, while the recomputation curve follows Eq. 4. Shaded regions indicate the preferred option for a single request without communication–computation overlap.

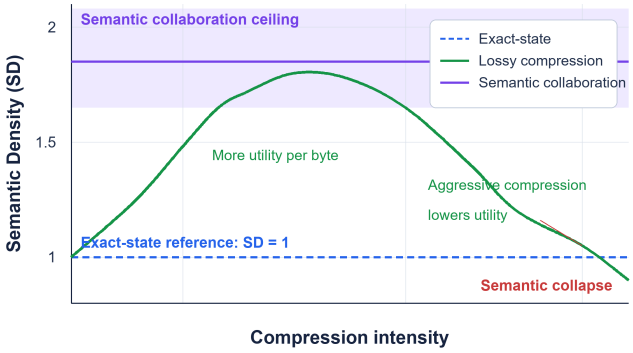

Fig. 5. Qualitative Semantic Density profiles of the three fidelity tiers (Eq. 2). The uncompressed exact-state reference is normalized to $\mathrm{SD} = 1$, while the other curves show illustrative trends rather than measured values.

For concurrent streams, pipelining, or overlapped transfer and computation, RTT may affect throughput and queueing rather than appear as a simple additive term. Eqs. 3 and 4 are therefore first order comparisons, not general performance models. When F is a KV cache, T_{comp} corresponds to one full prefill recomputation. When F is a hidden state or latent variable, it corresponds to the inference computation required to regenerate the representation.

In long context settings, the transfer decision remains sensitive to communication cost even when KV cache reduction is applied. Latent reasoning reduces the number of reasoning steps, but its latent vectors may still have dimensions comparable to the model hidden size. The per interaction payload is therefore not necessarily smaller than a text message. The objective of compression and reuse is not simply to minimize transferred bytes. It is to identify Pareto efficient operating points between transfer cost and recomputation cost under service level objectives (SLOs).

This trade-off is evident in prior work. FlexGen demonstrates the storage, computation, and communication trade-off by running OPT-175B on a single 16 GB GPU at about 1 token/s through tiered storage and large batch scheduling [67]. InfiniGen speculatively prefetches KV data according to attention scores during decoding, trading prefetching for reduced memory access latency [68]. Differentiable communication and latent message passing follow the same exchange logic. They transfer an intermediate representation once to avoid repeated reasoning [24], [25], [69].

Figure 4 visualizes this trade-off under representative parameters. The quadratic curve represents full prefill recomputation. The transfer curves use a 100 MB/s effective WAN bandwidth and a 200 ms RTT.

4.3 Information Gain and Bandwidth Cost

Transfer versus recomputation is only one Pareto axis. Vectorized communication also changes the task-relevant information available to downstream components. Reusing a KV state or latent state can avoid computation and reduce C_m. It can also preserve information that would be lost when a state is decoded into text, thereby increasing P_m. The cost is a larger or less compressible payload, which increases B_m and can reduce SD. SD captures this information-per-byte dimension, while L_m and C_m capture latency and recomputation cost.

The scenario analysis in Section 3.1 shows this dual effect. KV reuse removes repeated prefill over shared context, reducing B_m and C_m without necessarily reducing P_m. Latent reasoning replaces textual reasoning chains with continuous latent states and may reduce computation, but its effect on SD depends on both the task utility retained and the size of each latent vector. The KV volume follows Eq. 1 and depends on the number of KV heads, the attention architecture, and sequence length. For latent reasoning, fewer tokens do not guarantee fewer transferred bytes because each latent vector may have a dimension comparable to the model's hidden size.

Continuous representations can carry task-relevant information that is not preserved by discrete tokens, but their accuracy benefits depend on the task, model, latent-space alignment, and decoder compatibility. Latent-reasoning work (Coconut, SoftCoT, CODI) reports competitive accuracy with fewer reasoning steps in selected settings (see Section 6.3). N-gram embedding provides semantic enrichment by augmenting each token position with a local-context vector [58], [59], [60]. It enriches an internal representation and can still raise task utility per communicated byte. Figure 5 schematically summarizes the qualitative profiles of the three fidelity tiers. The uncompressed exact-state reference is normalized to $\mathrm{SD} = 1$, while the exact-state tier includes near-lossless methods whose SD may approach this reference. Lossy compression can increase SD over a moderate compression range before quality degrades, and semantic collaboration has a higher but more variable SD ceiling.

4.4 Exact-State Transfer Requirements

In PD-disaggregated architectures, the KV cache produced by prefill must reach the decode node before token generation begins. In SD terms, exact-state transfer aims to keep P_m close to the high-fidelity reference and therefore may have SD close to the reference value, but its large absolute

payload can still increase bandwidth consumption and latency. The core requirements are therefore timely delivery, low transfer volume, appropriate cache placement, and the ability to overlap communication with serving. Timely delivery is reflected in TTFT, while transfer volume determines bandwidth consumption and cache placement determines whether the required state is available at the destination. KV cache has therefore become an explicit workload spanning the network and storage boundary [46], [53].

Long-context scenarios exacerbate these requirements. Cross-data-center link bandwidth is far lower than intra-cluster high-speed interconnects, and KV cache transfer is constrained not only by bandwidth but also by RTT, which together determine TTFT. As context length grows, the absolute B_m and latency cost grow even when normalized SD remains close to the reference value; the system-level benefit of exact-state transfer therefore depends on whether reuse or computation avoidance justifies those bytes. Because ultra-long-context requests consume substantially more cache and transfer resources than ordinary requests, cross-wide-area systems should prioritize these high-cost requests when resources are limited.

Multi-agent scenarios impose an additional requirement. Cache reuse must remain correct when agents extend shared context differently. Correct reuse can reduce repeated B_m and C_m while preserving P_m, thereby increasing SD, but cross-agent offsets can corrupt the reused state and reduce task utility. These workloads typically involve extremely long contexts, short outputs, and substantial overlap across requests. KVCOMM's evaluation shows that the KV cache reuse rate on multi-agent workloads can exceed 70%, and that the core difficulty is the cache offset caused by cross-agent prefix divergence rather than transfer bandwidth itself [22].

The same reporting logic applies to cross-wide-area distributed training, where parameter and gradient exchanges recur over many optimization rounds rather than a one-time handoff. Evaluations should report communication volume per round and over the full training run, synchronization frequency, overlap with local computation, step time, time to target utility, and final task utility under specified bandwidth and RTT. The relevant latency is step time and time to target utility rather than TTFT. The reference horizon should be fixed by a target utility or a fixed compute budget, and both P_{ref} and B_{ref} should be measured over that horizon. Here, B_{ref} is the cumulative volume of the reference synchronization procedure, while B_m includes all cross-device bytes until method m reaches the same target. Communication-efficient local-update methods reduce B_m by synchronizing less often, but their operating point depends on convergence and final utility [28], [34].

4.5 Lossy Compression Transfer Requirements

For lossy compression, the SD objective is to reduce B_m while preserving enough task or generation quality to keep P_m high. A complete evaluation should report SD together with payload bytes, task or generation quality, encoding and decoding cost, end-to-end latency, and the network setting. These dimensions expose the trade-off among payload reduction, fidelity, overhead, and latency. Representative methods and results are reviewed in Section 6.1.

4.6 Semantic Collaboration Transfer Requirements

Semantic collaboration has a different requirement profile from exact-state transfer and lossy compression. Its SD objective is to maximize task utility P_m per transmitted byte B_m, rather than to preserve the original representation bit by bit. The transmitted representation must retain sufficient task-relevant information for downstream use, remain compatible with the receiving model, and satisfy payload and latency constraints. Evaluations should report SD together with task utility, payload bytes, end-to-end latency, model-alignment and compatibility conditions, and the fusion or decoding procedure. Representative methods and results are reviewed in Section 6.3.

4.7 Scenario Comparison

The three fidelity tiers represent different ways of navigating the SD–latency–recomputation Pareto frontier. Exact-state transfer prioritizes high P_m and reconstruction fidelity, potentially at the cost of large B_m. Lossy compression seeks to reduce B_m while keeping the resulting decrease in P_m acceptable. Semantic collaboration changes the communicated object and can improve P_m/B_m, but its utility depends more strongly on alignment and fusion. Relaxing reconstruction fidelity enlarges the optimization space, but can make task-level behavior less predictable. This distinction is reflected in the scenario matrix in Table 2.

5 Technical Challenges

This section identifies four technical challenges that limit progress along the SD–latency–recomputation Pareto frontier. Table 3 maps each challenge to its primary system layer.

5.1 Wide-Area Network Bottlenecks

Wide-area network conditions constrain vectorized communication through limited bandwidth, non-negligible RTT, and concurrent traffic. These constraints appear in three forms.

First, WAN capacity can fall far short of communication demand. NVLink provides several hundred Gbps, whereas regular cloud-server links typically provide single-digit Gbps [55]. Cross-cluster links can remain below 100 Gbps and remote KV-pool throughput below 10 Gbps, while one 32K-token Qwen3-235B configuration requires 2.1 Tbps of KV egress bandwidth [53]. KV communication can consequently account for 16%–60% of job completion time at 10–50 Gbps [53]. Second, RTT reduces effective throughput. RTTs increase from about 200 μs within a data center to about 10 ms between data centers and potentially hundreds of milliseconds across regions [76]. Geo-distributed KV movement can therefore remain RTT-bound even with abundant bandwidth [77]. Third, concurrent traffic makes the available rate variable. With up to 10 concurrent requests, a 1 GB KV stream can increase from 4 s at 2 Gbps to about 7 s when throughput falls to 0.2 Gbps [55]. These effects require adaptation to both network conditions and request concurrency.

TABLE 3
Mapping the four technical challenges to their primary system layers.

Challenge	Primary layer	Manifestation	Representative evidence
Wide-area network bottlenecks	Transport	Capacity gap, RTT sensitivity, shared-link contention	CacheGen [55], KVServe [53]
KV reuse limitations	Memory and reuse	Prefix matching, cross-agent offsets, non-prefix distortion, attention-pattern dependence, memory drift	KVCOMM [22], SGLang [70], SparseX [71], KVLink [72], CacheBlend [73], RedKnot [74], AgentKVShift [75]
Latent communication constraints	Representation and semantics	Cross-model misalignment, limited auditability, no unified distortion measure	Beyond Tokens [16], STAR-KV [51], CacheGen [55], CondenseFlow [61]
Cross-layer transfer interference	Cross-layer	Memory fragmentation, DMA inefficiency, head-of-line blocking	PagedAttention [9], CacheGen [55], CacheBlend [73]

5.2 KV Reuse Limitations

KV reuse reduces repeated transfer, but its effectiveness depends on structural compatibility between the reused context and the receiving computation. Existing mechanisms face six related limitations. First, prefix-based matching covers only exact prompt prefixes, leaving repeated content in multi-turn dialogue, retrieval-augmented generation, and multi-agent interaction outside the reuse scope [70]. Second, independently extended contexts create cross-agent positional offsets. Overlapping content may therefore refer to different positions, making direct reuse unsafe [22]. Third, non-prefix reuse must preserve both positional information and cross-segment attention. A duplicated segment may not retain the same attention behavior after relocation, so reuse can require selective corrective recomputation rather than simple cache concatenation [71], [72], [73].

Reuse is also sensitive to the underlying attention pattern. Different mechanisms impose different constraints on cache layout, valid reuse regions, and correction cost [74]. Agent memory changes across dialogue turns, which can make static cache mappings stale and require continual validation or refresh [75]. Finally, cross-model reuse faces dimension and representation incompatibility. States produced by different architectures generally cannot be consumed directly without an alignment mechanism, although recent work explores linear mappings for compatible model families [16], [78]. Together, these limitations show that reuse granularity, positional consistency, attention-pattern compatibility, temporal freshness, and model compatibility must be handled jointly.

5.3 Latent Communication Constraints

As an emerging paradigm, latent communication faces several technical gaps. First, different model architectures may use different numbers of layers, hidden dimensions, and attention mechanisms. Their latent spaces therefore lack a natural alignment, and representations transferred between models cannot be directly interpreted [16]. Second, latent communication raises security concerns. Unlike natural-language messages, continuous representations are difficult to audit or interpret, and conventional content-safety filters cannot operate on them directly (see Section 7.2). Third, latent compression must balance information fidelity and transfer efficiency. Existing work often evaluates this balance indirectly through task performance and lacks a unified distortion measure [51], [55], [61].

5.4 Cross-Layer Transfer Interference

Existing compression and transfer algorithms generally assume that KV caches are physically contiguous in memory, an assumption that rarely holds in production systems. Inference engines such as PagedAttention manage KV caches in fixed token blocks (16 tokens per block by default in vLLM), and physical blocks are non-contiguous in memory [9]. When transferring across wide-area networks, this fragmentation causes two forms of system-level coupling interference.

The first is PCIe/DMA efficiency collapse. If non-contiguous pages are transferred directly via RDMA or sockets without memory compaction, the NIC must perform many gather-scatter DMA operations, reducing effective PCIe bandwidth. If memory is compacted first, the extra GPU/CPU copies compete for the memory bandwidth of the decode phase and may reduce throughput instead of improving it. The second is link head-of-line blocking. On a shared wide-area line, a very large KV cache request can occupy the link for a long time and block subsequent small requests. This is both a bandwidth contention problem and a tail-latency problem caused by the absence of transfer scheduling. Together, these effects show that cross-wide-area transfer optimization must be co-designed with inference-engine memory management and NIC transfer semantics [9], [55], [73].

6 Optimization Routes for Vectorized Communication

The optimization routes in this section seek better points on the SD–latency–recomputation Pareto frontier by reducing transmitted bytes, increasing task utility per byte, lowering end-to-end latency, or avoiding redundant recomputation. Table 4 summarizes the four routes along their objectives, mechanisms, representative methods, and evidence scope. The routes can be combined into hybrid policies under service-level objectives. Across the route sections, we record the communication object, fidelity tier, system boundary, workload, baseline, and primary metric whenever the source reports them. Table 5 gives a compact comparison of representative systems.

TABLE 4
Overview of the four optimization routes in Section 6.

Route	Objective	Mechanism	Object	Representative systems	Evidence scope
Bandwidth compression (Sec. 6.1)	Transfer less	Quantization, low-rank, selective, lossless	Compressed KV	KIVI, KVQuant, ZipCache, TurboQuant, STAR-KV, SmartGen, CacheGen, KVServe, SplitZip [49], [50], [51], [53], [54], [55], [56], [64], [65]	Mostly memory/transfer prototypes, WAN evidence varies
Reuse & caching (Sec. 6.2)	Transfer nothing	Prefix/segment reuse, relay, cache layers	Reused KV	SGLang, Preble, SparseX, KVLink, CacheBlend, HYPIC, LinearKV, RedKnot, AgentKVShift, PDD, LMCache, KVCOMM [4], [22], [46], [47], [70], [71], [72], [73], [74], [75], [79], [80]	Serving systems, gains depend on reuse patterns
Latent & semantic communication (Sec. 6.3)	Transfer semantics	Latent reasoning, message passing, semantic compression	Latent states / representations	Coconut, SoftCoT, CODI, LatentMAS, Interlat, Agent Primitives, CondenseFlow [19], [20], [21], [24], [25], [26], [61]	Task- and model-specific research
Storage & prefetch (Sec. 6.4)	Transfer later	Tiering plus on-demand prefetch	Staged KV	FlexGen, InfiniGen, Mooncake, LMCache [3], [46], [67], [68]	Storage/serving prototypes, WAN evidence limited

TABLE 5
Compact comparison of representative systems.

Route / work	Object	Scenario	Primary metric	Reported result	Evidence scope
Compression: CacheGen [55]	KV bitstream	Online KV transfer	Bandwidth	3.5–4.3× bandwidth saving	Near-lossless transfer prototype
Reuse: KVCOMM [22]	KV cache	Cross-agent reuse	Reuse / speedup	Cache reuse $> 70\%$, speedup up to 7.8×	Five-agent workload
Relay: PDD [4]	KV plus token IDs	Cross-data-center PD	P90 TTFT / payload	P90 TTFT 18.3 s $\rightarrow$ 9.8 s, payload 4,649 KB $\rightarrow$ 1.5 KB	64K context, 90% prefix hit
Latent: LatentMAS [24]	Last-layer embeddings	Latent multi-agent	Accuracy / tokens	Accuracy improvement up to 14.6%, output-token reduction 70.8–83.7%	Nine benchmarks, task-specific
Semantic compression: CondenseFlow [61]	Fixed-length semantic KV	Semantic compression	Memory / latency / score	KV-memory reduction $> 99\%$, latency reduction about 20%, score gain 1.7 points	Seven benchmarks, model/task specific
Storage: FlexGen [67]	Tiered model/KV storage	Storage offload	Throughput	OPT-175B on a 16 GB GPU at about 1 token/s	Single-GPU system prototype

6.1 Bandwidth Compression

Bandwidth compression reduces transfer time directly by lowering the number of bits per unit of information and is currently the most widely studied route. It can be divided into five families. These are quantization, low-rank decomposition, service-aware adaptive compression, selective transfer, and lossless compression.

From the SD perspective, this route reduces B_m and can increase SD when task utility P_m is preserved. Stronger compression may reduce P_m or add encoding overhead, so its operating point depends on fidelity and latency requirements.

Quantization compresses BF16 KV caches to lower bit widths. KIVI proposes tuning-free asymmetric 2-bit quantization and is a representative baseline for later quantization work [49]. KVQuant designs per-channel and per-token low-bit schemes for outlier distributions [64]. ZipCache combines salient-token identification with mixed-precision quantization [65]. PolarQuant applies random preconditioning and polar transformation to reduce normalization overhead in low-bit KV quantization [81]. TurboQuant provides online vector quantization with near-optimal distortion and reports quality-neutral KV-cache quantization at 3.5 bits per channel [50]. Since KV compression reduces the bytes that a link must carry for the same context, it reduces bandwidth demand and transfer time in addition to GPU memory use. In the SD formulation, this lowers B_m and raises SD when P_m is preserved, provided that encoding and decoding overhead remain limited.

Low-rank decomposition exploits redundancy along the hidden dimension. STAR-KV uses a differentiable soft threshold for adaptive rank selection, applies different decomposition sensitivity strategies to Keys and Values, and combines mixed-precision quantization to achieve up to 20× compression and a 3.1× end-to-end throughput improvement [51].

Service-aware adaptive compression links compression policy with service-level objectives. KVServe unifies KV

compression into a modular policy space, searches for configurations offline with a Bayesian analysis engine, and dynamically selects compression levels online with a controller that combines an analytical latency model and a lightweight bandit algorithm. Compression is tightened when bandwidth is scarce, and light compression or no compression is chosen when bandwidth is ample to avoid overhead [53]. The cited work illustrates joint compression-transfer optimization.

Selective transfer breaks the implicit assumption that all KV entries are equally important. SmartGen pre-pushes the KV of universally important tokens based on offline profiling, pulls context-dependent tokens in parallel during decode, and transfers secondary tokens on idle bandwidth. This minimizes the amount transferred on the critical path. TTST is reduced by up to 4.3× relative to full KV transfer while generation quality stays near-lossless [54].

Head-aware compression further adapts the transfer budget to heterogeneous attention behavior. HybridKV classifies attention heads as static or dynamic and applies text-prior pruning or chunk-wise retrieval accordingly, illustrating how head-level structure can reduce KV volume for multimodal LLM serving [52].

Lossless compression targets scenarios that must recover bits exactly. SplitZip exploits frequency redundancy in BF16 floating-point exponent values with fixed-length encoding of high-frequency exponents, achieving GPU-friendly online lossless compression with 613.3 GB/s compression and 2181.8 GB/s decompression throughput, substantially better than prior lossless compressors for latency-critical paths [56]. CacheGen's streaming arithmetic coding provides 3.5–4.3× bandwidth savings under near-lossless constraints [55].

Quantization and low-rank methods can achieve high compression ratios but require model- and task-specific fidelity evaluation. Service-aware methods target system-level gains. Selective transfer and lossless compression address narrower operating conditions.

6.2 Reuse and Caching

KV reuse reduces transfer volume by transmitting less data or avoiding transfer altogether. Its evolution has progressed from prefix reuse to arbitrary-segment reuse, relay transfer, and system-level cache layers, steadily expanding the optimization space for cross-wide-area transfer.

Exact reuse can reduce both B_m and C_m without lowering P_m, whereas non-exact reuse may trade recomputation against task utility.

Prefix reuse is the earliest and most established form. Requests sharing the same prompt prefix reuse cached KV, avoiding repeated prefill computation and transfer. Typical implementations include vLLM's automatic prefix caching [9] and SGLang's RadixAttention [70]. PDD further proposes Decode-side radix caching (DRC) that caches historical KV at decode instances, reducing cross-data-center bandwidth demand by up to 10× at a 90% average prefix hit rate [4].

Non-prefix reuse extends the reuse unit from prefixes to arbitrary continuous segments. SparseX uses token segments as the reuse unit, a Sparse-Q index to identify key tokens that need corrective recomputation, and layer-wise mixed attention. It uses full computation in early layers and sparse recomputation in later layers to recover cross-segment interactions [71]. KVLink addresses the attention distortion of non-prefix concatenation through positional re-encoding and inter-document link tokens [72]. CacheBlend selectively recomputes key tokens at concatenation points to recover cross-segment attention at low cost [73].

Hybrid-attention models add another reuse constraint because different layers follow different state-transition rules. HYPIC extends position-independent caching to hybrid-attention models by composing cached transition operators for linear-attention layers and applying local recomputation around concatenation boundaries for full-attention layers [79]. LinearKV instead uses a single cached state to initialize linear-attention layers and retains token-level reuse for full-attention layers, giving a contrasting design for hybrid-attention position-independent caching [80].

Dedicated designs have also emerged for attention-head heterogeneity and agent memory. RedKnot proposes a head-aware KV cache management system with head-level paging and selective recomputation. It reduces TTFT by up to 12× in long-context serving [74]. AgentKVShift addresses dynamic shifts in agent memory through probe-guided KV residual correction. It reaches near-full-recompute performance while refreshing only 10%–30% of the cache and speeds up prefill by 2–3.5× [75].

Relay transfer masks wide-area latency with a *relay + recompute* hybrid strategy. PDD inserts a RelayDecode (RLD) instance between prefill and decode. RLD is co-located with prefill in the same domain. It receives the KV within milliseconds after prefill completes and starts decoding the first token. This masks the cross-data-center KV transfer latency. Once the main decode (MD) instance is ready, RLD sends only lightweight token IDs. MD then recomputes the incremental KV locally and takes over. On an agent workload with 64K average context and a 90% prefix hit rate, PDD reduces P90 TTFT from 18.3 s to 9.8 s, a 46% reduction. RLD generates only 6.2% of the total output tokens. The handover payload per 100 tokens decreases from 4,649 KB to 1.5 KB. The benefit-cost ratio improves by up to 37.5% over a homogeneous same-domain PD baseline [4].

At the infrastructure layer, system-level cache sharing extends KV reuse beyond individual requests and engines. LMCache decouples the KV cache from GPU memory and provides cross-engine, cross-query cache offloading and sharing. It improves throughput by up to 15× when combined with vLLM [46]. KVCOMM achieves over 70% cross-agent cache reuse with up to 7.8× speedup in multi-agent scenarios [22].

The overall trend is that reuse granularity moves from prefixes to segments, applicable scenarios move from single-node to multi-agent and cross-cluster settings, and reuse gains are closely related to workload repetition.

6.3 Latent-Space Reasoning and Vectorized Interaction

Latent-space communication shifts multi-agent collaboration from transferring data to transferring semantics. Its origin can be traced to differentiable communication in multi-agent reinforcement learning. Foerster et al. [69] proposed

an end-to-end differentiable framework in which communication protocols are learned through gradient training. With LLMs serving as general reasoning engines, communication content has expanded from control signals to knowledge representations. Semantic communication provides a useful reference for quantifying this shift [11].

From the SD perspective, this route directly targets the ratio P_m/B_m, but its achievable point depends on representation alignment, fusion, and task utility.

The shift from text-based to vectorized interaction is key to this route. Text-based multi-agent frameworks such as ReAct [8] and AutoGen [7] pass reasoning traces through natural language, which is interpretable and auditable but incurs three structural costs. These costs include the extra reasoning cost of text generation and parsing, irreversible information loss from discretization, and error accumulation and information decay across long interaction rounds. Vectorized interaction passes information through internal model representations and avoids these problems [16].

At the design-pattern level, Agent Primitives borrows the idea of reusable neural-network components. It decomposes multi-agent architectures into three high-frequency patterns. These are Review, Voting & Selection, and Planning & Execution. It passes review context, option comparison, and planning intent through KV caches. The method improves accuracy by 12.0–16.5 percentage points over a single-agent baseline while reducing token usage and inference latency [26].

At the pure latent collaboration level, LatentMAS proposes a training-free framework in which agents reason autoregressively over last-layer hidden embeddings and maintain cross-agent internal state through a shared latent working memory, achieving up to 14.6% accuracy improvement and 70.8%–83.7% output token reduction across nine benchmarks [24]. Interlat communicates through the last hidden state directly as a continuous representation of *thought*, is simple and training-free, and achieves up to 24× inference speedup on long-context tasks [25].

Latent reasoning provides a complementary route for vectorized interaction. Rather than relying on inter-agent message passing, the model completes part of its reasoning in continuous latent space. Coconut replaces textual reasoning steps with continuous latent states and reports gains on logic tasks requiring backtracking, with fewer thinking tokens [19]. SoftCoT uses a small model to generate continuous soft thinking tokens that guide the main model [20]. CODI distills explicit chain-of-thought into latent space and reports a 3.1× compression ratio at GPT-2 scale [21]. These results suggest that latent communication can improve the accuracy-efficiency trade-off in selected settings, but the results remain model- and task-dependent. Knowledge distillation provides a related channel for vectorized knowledge transfer. Hinton et al. [45] compress teacher knowledge into a student model, suggesting one possible mechanism for future cross-model collaboration.

At the semantic compression level, CondenseFlow compresses KV caches into fixed-length representations via learnable semantic probes. This decouples communication complexity from context length. Across seven benchmarks, it reduces KV memory usage by over 99% and outperforms text-based methods by an average of 1.7 points [61].

At the unified-framework level, Beyond Tokens classifies 18 representative works along three axes (communication content, alignment, and fusion), identifies five design patterns, and reports latency reductions of 2–24× relative to natural-language communication. It also identifies two recurring gaps. These are cross-architecture alignment and the lack of standardized evaluation [16].

Overall, this route is still in its early stage. Most methods target specific tasks and models, and training requirements, communication media, and evaluation protocols differ. Cross-method comparison therefore lacks a unified benchmark.

6.4 Hierarchical Storage and Remote Prefetching

Beyond compression and semantic abstraction, deployed systems also treat KV caches as *hot/cold pages*, placing them in DRAM, NVMe SSD, or remote object storage and prefetching them during decode. The key indicator for this route is cache hit rate rather than compression ratio.

Unlike compression and reuse, storage and prefetch primarily shift when and where transfer occurs. Their main effect is on critical-path L_m and redundant C_m, while their SD impact is indirect through cache hits and transfer avoidance.

FlexGen examined the storage-compute trade-off by running large models on a single GPU through tiered storage and large-batch scheduling [67]. InfiniGen speculatively prefetches KV data according to attention scores during decode, overlapping memory access with computation, and reports up to about 3× speedup [68]. Mooncake makes the hierarchical KV pool part of its disaggregated architecture, using CPU, DRAM, and SSD resources to host KV caches so that prefill and decode clusters share a cache view [3]. LMCache abstracts the cache layer as cross-engine, cross-query infrastructure for cache offloading and cross-GPU transfer [46].

In cross-wide-area scenarios, when multiple requests share a long system prompt, remote storage can keep one base KV copy and allow later requests to reuse it. Storage I/O may be faster than a wide-area round trip, but storage bandwidth is also limited and cache misses are costly. A practical system therefore needs to balance recomputation, compression, and caching according to network load, GPU utilization, and storage IOPS [3], [46], [67], [68].

7 Open Problems and Future Directions

This section discusses seven open problems in cross-wide-area vectorized transfer. They span representation alignment, system support, evaluation, economics, and communication beyond current Transformer architectures. Together, they determine how effectively future systems can navigate the SD–latency–recomputation trade-off.

7.1 Cross-Architecture Alignment

Latent spaces from different LLM architectures lack a shared coordinate system, limiting cross-model communication. Differences in layer structure, hidden dimensions, and attention mechanisms make direct exchange unreliable and can reduce task utility P_m. This uncertainty also makes

the semantic density of a transferred representation difficult to predict across models. Future work needs architecture-agnostic aligners, compatibility protocols, and measurable alignment criteria, while recent linear-mapping methods provide an initial path for compatible model families [16], [78].

7.2 Latent-Channel Security and Auditability

Continuous latent representations are difficult to interpret, filter, and audit. This limits the ability to inspect transmitted content, detect unsafe information, and identify the source of harmful behavior in multi-agent workflows. Security mechanisms must therefore operate on latent channels without eliminating their efficiency and task-utility benefits. Future work should develop latent-channel protection, provenance tracking, and reversible auditing methods that preserve communication efficiency.

7.3 Compression-Fidelity Trade-off

Compression must balance information fidelity, transfer efficiency, and encoding overhead. The useful operating point depends on task utility P_m, payload size B_m, encoding and decoding cost, and end-to-end latency. Excessive compression can reduce P_m and semantic density, while insufficient compression wastes bandwidth and may increase transfer latency. Existing work lacks a unified distortion measure and a transferable model of the compression–task utility relationship [51], [53], [55].

7.4 Standardization and Evaluation

KV caches and latent representations lack common formats and evaluation protocols. Different layouts, paging granularities, tensor precisions, and alignment assumptions make cross-system reuse and transfer costly [9], [70]. Evaluation is also fragmented across models, tasks, fidelity levels, and network conditions, which obscures the relationship between SD, latency, and recomputation. Future work should standardize cache and representation interfaces and establish testbeds spanning models, tasks, workloads, and network conditions [16], [82].

7.5 Economics of Wide-Area Transfer

Cross-region traffic can make transfer more expensive than recomputation. The decision depends on traffic pricing, GPU utilization, cache reuse, and the latency benefit of moving a representation instead of regenerating it. These factors determine whether a transfer policy improves the system-level Pareto point even when it reduces computation. Future systems should jointly optimize compute, bandwidth, cache sharing, and service-level objectives under explicit traffic costs [10], [82].

7.6 End-to-End Transferability

Model architecture and communication are still optimized largely independently. As a result, internal states may be expensive to serialize, difficult to reuse, or poorly suited to cross-device transfer. Future models should expose representations that are easier to compress, reuse, and transfer through low-rank, sparse, or separable attention structures. This direction requires co-design between model architecture, inference engines, and network interfaces.

7.7 Beyond Transformers

State-space, non-autoregressive, and diffusion models change the communicated object beyond the KV cache [83], [84], [85], [86]. Their state streams, intermediate representations, or noise variables may have different reuse, compression, and alignment properties. Future taxonomies and protocols should therefore cover these objects while retaining the trade-off between data movement, latency, and recomputation. This broader view will determine whether vectorized communication remains centered on KV transfer or evolves toward more general task-state exchange.

8 Conclusion

By treating parameters, activations, KV caches, hidden states, and latent messages as vectorized communication objects, this survey has brought distributed training, disaggregated inference, KV pooling, and multi-agent interaction into a common framework. The framework relates representation fidelity to task utility, transferred bytes, end-to-end latency, and redundant recomputation. Its central analytical tool, Semantic Density, reports task utility per normalized byte relative to an uncompressed exact-state reference, while retaining latency and recomputation as separate system costs. This perspective has mapped the design space across usage, fidelity, and optimization route. It has also exposed the recurring systems issues that limit cross-wide-area deployment. The choice among transfer, compression, reuse, storage, and recomputation depends on the utility preserved and the resulting network and computation costs. Future work should connect model architecture, communication protocols, storage systems, and evaluation standards, with particular attention to cross-model alignment, auditable latent channels, training-aware communication metrics, and state streams beyond KV caches. Vectorized communication is therefore best viewed as a controllable way to exchange task-relevant state across distributed LLM systems rather than as a universal replacement for text or computation.

References

[1] P. Patel, E. Choukse, C. Zhang *et al.*, "Splitwise: Efficient generative LLM inference using phase splitting," in *Proceedings of the 51st International Symposium on Computer Architecture (ISCA)*, Buenos Aires, Argentina, 2024, pp. 118–132.

[2] Y. M. Zhong, S. Y. Liu, J. D. Chen *et al.*, "DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving," in *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Santa Clara, CA, USA, 2024, pp. 193–210.

[3] R. Y. Qin, Z. X. Li, W. He *et al.*, "Mooncake: A KVCache-centric disaggregated architecture for LLM serving," *arXiv preprint arXiv:2407.00079*, 2024.

[4] Y. Wang, X. Li, J. Ma, G. Sun, Y. Xu, B. Han, J. Ng, Y. Luo, K. Hong, G. Dai, B. Li, and Y. Wang, "PDD: Unleashing economical and flexible heterogeneous LLM inference via cross-datacenter prefill-decode disaggregation," Infinigence-AI, Technical report, 2026. [Online]. Available: https://github.com/infinigence/pdd

[5] R. Zhu, Z. Jiang, C. Jin, P. Wu, C. A. Stuardo, D. Wang, X. Zhang, H. Zhou, H. Wei, Y. Cheng, J. Xiao, X. Zhang, L. Lin, L.-W. Chang, J. Ye, X. Yu, X. Liu, X. Jin, and X. Liu, "MegaScale-Infer: Serving mixture-of-experts at scale with disaggregated expert parallelism," *arXiv preprint arXiv:2504.02263*, 2025. [Online]. Available: https://arxiv.org/abs/2504.02263

[6] H. Wu, A. R. Bambhaniya, S. Banerjee, T. Khare, S. Srinivasan, S. Kundu, M. Elavazhagan, W. Won, A. Yazdanbakhsh, and T. Krishna, "How far can disaggregation go? a design-space exploration of attention-FFN disaggregation for efficient MoE LLM serving," *arXiv preprint arXiv:2605.28302*, 2026. [Online]. Available: https://arxiv.org/abs/2605.28302

[7] Q. Y. Wu, G. Bansal, J. Y. Zhang *et al.*, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation framework," *arXiv preprint arXiv:2308.08155*, 2023.

[8] S. Y. Yao, J. Zhao, D. Yu *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 2023.

[9] W. Kwon, Z. Li, S. Y. Zhuang *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*, New York, NY, USA, 2023.

[10] N. Development, R. Commission, O. of the Central Cyberspace Affairs Commission, M. of Industry, I. Technology, and N. E. Administration, "Reply on approving the construction of national integrated computing power network hub nodes in the Beijing-Tianjin-Hebei, Yangtze River Delta, Guangdong-Hong Kong-Macao Greater Bay Area, Chengdu-Chongqing, Inner Mongolia, Guizhou, Gansu, and Ningxia regions," Policy document, Beijing, China, 2022.

[11] H. Q. Xie, Z. J. Qin, G. Y. Li *et al.*, "Deep learning enabled semantic communication systems," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2663–2675, 2021.

[12] S. P. Hu, G. Y. Zhang, and W. M. Zheng, "Survey on KV cache compression methods for large language model inference," *Journal of Computer Research and Development*, vol. 63, no. 4, pp. 1021–1038, 2026.

[13] H. Li, Y. Li, A. Tian, T. Tang, Z. Xu, X. Chen, N. Hu, W. Dong, L. Qing, and L. Chen, "A survey on large language model acceleration based on KV cache management," *Transactions on Machine Learning Research*, 2025. [Online]. Available: https://mlanthology.org/tmlr/2025/li2025tmlr-survey/

[14] J. Pan and G. Li, "A survey of LLM inference systems," *arXiv preprint arXiv:2506.21901*, 2025. [Online]. Available: https://arxiv.org/abs/2506.21901

[15] Y. Y. Hu, S. C. Liu, Y. W. Yue *et al.*, "Memory in the age of AI agents," *arXiv preprint arXiv:2512.13564*, 2025.

[16] Y. Z. Liu, "Beyond tokens: A unified framework for latent communication in LLM-based multi-agent systems," *arXiv preprint arXiv:2606.05711*, 2026.

[17] B. Yan, X. Zhang, L. Zhang, L. Zhang, Z. Zhou, D. Miao, and C. Li, "Beyond self-talk: A communication-centric survey of llm-based multi-agent systems," *arXiv preprint arXiv:2502.14321*, 2025. [Online]. Available: https://arxiv.org/abs/2502.14321

[18] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, Long Beach, CA, USA, 2017, pp. 5998–6008.

[19] S. B. Hao, S. Sukhbaatar, D. J. Su *et al.*, "Training large language models to reason in a continuous latent space," *arXiv preprint arXiv:2412.06769*, 2024.

[20] Y. G. Xu, X. Guo, Z. W. Zeng *et al.*, "SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Vienna, Austria, 2025, pp. 23 336–23 351.

[21] Z. Y. Shen *et al.*, "CODI: Compressing chain-of-thought into continuous space via self-distillation," *arXiv preprint arXiv:2502.21074*, 2025.

[22] H. C. Ye, Z. Q. Gao, M. Y. Ma *et al.*, "KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems," in *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025, arXiv:2510.12872.

[23] Y. S. Geng, Y. C. Gao, W. H. Wu *et al.*, "RelayCaching: Accelerating LLM collaboration via decoding KV cache reuse," *arXiv preprint arXiv:2603.13289*, 2026.

[24] J. R. Zou, X. Y. Yang, R. Z. Qiu *et al.*, "Latent collaboration in multi-agent systems," *arXiv preprint arXiv:2511.20639*, 2025.

[25] Z. Y. Du, R. Z. Wang, H. Y. Bai *et al.*, "Enabling agents to communicate entirely in latent space," in *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, San Diego, CA, USA, 2026, pp. 27 106–27 129.

[26] H. B. Jin, P. Kuang, Y. Yu *et al.*, "Agent primitives: Reusable latent building blocks for multi-agent systems," *arXiv preprint arXiv:2602.03695*, 2026.

[27] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3-4, pp. 379–423, 623–656, 1948.

[28] A. Douillard, Q. Feng, A. A. Rusu *et al.*, "DiLoCo: Distributed low-communication training of language models," *arXiv preprint arXiv:2311.08105*, 2023.

[29] A. Douillard, Q. Feng, A. A. Rusu, A. Kuncoro, Y. Donchev, R. Chhaparia, I. Gog, M. Ranzato, J. Shen, and A. Szlam, "DiPaCo: Distributed path composition," *arXiv preprint arXiv:2403.10616*, 2024. [Online]. Available: https://arxiv.org/abs/2403.10616

[30] S. Jaghouar, J. M. Ong, and J. Hagemann, "OpenDiLoCo: An open-source framework for globally distributed low-communication training," *arXiv preprint arXiv:2407.07852*, 2024. [Online]. Available: https://arxiv.org/abs/2407.07852

[31] L. Sani, A. Iacob, Z. Cao, R. Lee, B. Marino, Y. Gao, D. Cai, Z. Li, X. Qiu, and N. D. Lane, "Photon: Federated LLM pre-training," *arXiv preprint arXiv:2411.02908*, 2024. [Online]. Available: https://arxiv.org/abs/2411.02908

[32] A. Douillard, K. Rush, Y. Donchev, Z. Charles, N. Fallen, A. Dubey, I. Gog, J. Dean, B. Woodworth, Z. Garrett, N. Keating, J. Bishop, H. Prior, E. Yvinec, A. Szlam, M. Ranzato, and J. Dean, "Decoupled DiLoCo for resilient distributed pre-training," *arXiv preprint arXiv:2604.21428*, 2026. [Online]. Available: https://arxiv.org/abs/2604.21428

[33] G. Avraham, V. Shevchenko, H. M. Dolatabadi, K. Pajak, J. Snewin, H. Xi, R. O'Donnell, T. Ajanthan, S. Ramasinghe, C. H. Koneputugodage, S. Siriwardhana, and A. Long, "Agora: Collective and permissionless internet-scale pretraining of large language models," *arXiv preprint arXiv:2607.13332*, 2026. [Online]. Available: https://arxiv.org/abs/2607.13332

[34] Z. Charles, G. Teston, L. M. Dery *et al.*, "Communication-efficient language model training scales reliably and robustly: Scaling laws for DiLoCo," in *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025, arXiv:2503.09799.

[35] H. J. An, S. D. Huang, S. Q. Huang *et al.*, "AI Flow: Perspectives, scenarios, and approaches," *arXiv preprint arXiv:2506.12479*, 2025.

[36] W. J. Chu *et al.*, "A cloud-edge collaborative inference system for data-secure LLM serving," in *Proceedings of the 2nd ACM Workshop on Networks for AI Computing (NetAI)*, 2025.

[37] P. Y. Zhu and T. T. Yang, "CE-LSLM: Efficient large-small language model inference and communication via cloud-edge collaboration," *arXiv preprint arXiv:2505.14085*, 2025.

[38] Y. Jin, T. Wang, H. Lin *et al.*, "P/D-Serve: Serving disaggregated large language model at scale," *arXiv preprint arXiv:2408.08147*, 2024. [Online]. Available: https://arxiv.org/abs/2408.08147

[39] S. Chen, R. Jiang, D. Yu, J. Xu, M. Chao, F. Meng, C. Jiang, W. Xu, and H. Liu, "KVDirect: Distributed disaggregated LLM inference," *arXiv preprint arXiv:2501.14743*, 2025. [Online]. Available: https://arxiv.org/abs/2501.14743

[40] R. Y. Qin, W. R. He, Y. Y. Wang *et al.*, "Prefill-as-a-service: KV-Cache of next-generation models could go cross-datacenter," *arXiv preprint arXiv:2604.15039*, 2026.

[41] W. Li, G. Jiang, X. Ding, Z. Tao, C. Hao, C. Xu, Y. Zhang, and H. Wang, "FlowKV: A disaggregated inference framework with low-latency KV cache transfer and load-aware scheduling," *arXiv preprint arXiv:2504.03775*, 2025. [Online]. Available: https://arxiv.org/abs/2504.03775

[42] Alibaba Cloud, "Tair KVCache Manager: Architecture design and implementation of an enterprise-grade global KV-Cache management service," Alibaba Cloud Developer, 2025, https://developer.aliyun.com/article/1703191.

[43] C. Packer, S. Wooders, K. Lin *et al.*, "MemGPT: Towards LLMs as operating systems," *arXiv preprint arXiv:2310.08560*, 2023.

[44] W. J. Zhong, L. H. Guo, Q. Q. Gao *et al.*, "MemoryBank: Enhancing large language models with long-term memory," in *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI)*, vol. 38, no. 17, 2024, pp. 19 724–19 731.

[45] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[46] Y. H. Cheng, Y. H. Liu *et al.*, "LMCache: An efficient KV cache layer for enterprise-scale LLM inference," *arXiv preprint arXiv:2510.09665*, 2025.

[47] V. Srivatsa, Z. J. He, R. Abhyankar *et al.*, "Preble: Efficient distributed prompt scheduling for LLM serving," in *Proceedings of*

the 13th International Conference on Learning Representations (ICLR), 2025.

[48] H. Dong, T. Johnson, M. Cho, and E. Soroush, "Towards low-bit communication for tensor parallel LLM inference," *arXiv preprint arXiv:2411.07942*, 2024. [Online]. Available: https://arxiv.org/abs/2411.07942

[49] Z. R. Liu, J. Y. Yuan, H. Y. Jin *et al.*, "KIVI: A tuning-free asymmetric 2bit quantization for KV cache," in *Proceedings of the 41st International Conference on Machine Learning (ICML)*, vol. 235. PMLR, 2024, pp. 32 332–32 344.

[50] A. Zandieh, M. Daliri, M. Hadian, and V. Mirrokni, "TurboQuant: Online vector quantization with near-optimal distortion rate," *arXiv preprint arXiv:2504.19874*, 2025.

[51] P. Bhatnagar, A. Moradifirouzabadi, S. H. Yang *et al.*, "STAR-KV: Low-rank KV cache compression via soft thresholding for adaptive rank control," *arXiv preprint arXiv:2606.08382*, 2026.

[52] B. Zeng, F. Ren, J. Zhang, X. Gu, K. Chen, L. Shou, and H. Li, "HybridKV: Hybrid KV cache compression for efficient multimodal large language model inference," *arXiv preprint arXiv:2604.05887*, 2026. [Online]. Available: https://arxiv.org/abs/2604.05887

[53] Z. D. Liu, X. Y. Ma, D. J. Luo *et al.*, "KVServe: Service-aware KV cache compression for communication-efficient disaggregated LLM serving," *arXiv preprint arXiv:2605.13734*, 2026, accepted at ACM SIGCOMM 2026.

[54] X. C. Luo, J. C. Shen, X. Wang *et al.*, "SmartGen: Seamless disaggregated LLM inference with selective KV cache transfer," *arXiv preprint arXiv:2607.28150*, 2026.

[55] Y. H. Liu, H. C. Li, Y. H. Cheng *et al.*, "CacheGen: KV cache compression and streaming for fast large language model serving," in *Proceedings of the ACM SIGCOMM 2024 Conference*, Sydney, Australia, 2024.

[56] Y. P. Guo and S. Joshi, "SplitZip: Ultra fast lossless KV compression for disaggregated LLM serving," *arXiv preprint arXiv:2605.01708*, 2026.

[57] B. Kriuk and L. Ng, "Q-KVComm: Efficient multi-agent communication via adaptive KV cache compression," *arXiv preprint arXiv:2512.17914*, 2025.

[58] H. Liu, J. Zhang, C. Wang, X. Hu, L. Lyu, J. Sun, X. Yang, B. Wang, F. Li, Y. Qian, L. Si, Y. Sun, R. Li, P. Pei, Y. Xie, and X. Cai, "Scaling embeddings outperforms scaling experts in language models," *arXiv preprint arXiv:2601.21204*, 2026. [Online]. Available: https://arxiv.org/abs/2601.21204

[59] X. Cheng, W. Zeng, D. Dai, Q. Chen, B. Wang, Z. Xie, K. Huang, X. Yu, Z. Hao, H. Zhang, Y.-K. Li, H. Zhang, D. Zhao, and W. Liang, "Conditional memory via scalable lookup: A new axis of sparsity for large language models," in *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2026, pp. 4968–4990. [Online]. Available: https://aclanthology.org/2026.acl-long.226/

[60] Qwen Team, "Qwen3.8-Flash-Next," Model card, 2026. [Online]. Available: https://huggingface.co/Qwen/Qwen3.8-Flash-Next

[61] X. Y. Chen, F. G. Wu, J. S. Zhao *et al.*, "CondenseFlow: Scalable latent space collaboration via semantic compression for multi-agent systems," in *Findings of the Association for Computational Linguistics: ACL 2026*, San Diego, CA, USA, 2026, pp. 13 694–13 712.

[62] T. Ji, B. Guo, Y. Wu *et al.*, "Towards economical inference: Enabling DeepSeek's multi-head latent attention in any Transformer-based LLMs," *arXiv preprint arXiv:2502.14837*, 2025. [Online]. Available: https://arxiv.org/abs/2502.14837

[63] DeepSeek-AI *et al.*, "DeepSeek-V4: Towards highly efficient million-token context intelligence," *arXiv preprint arXiv:2606.19348*, 2026. [Online]. Available: https://arxiv.org/abs/2606.19348

[64] C. Hooper, S. Kim, H. Mohammadzadeh *et al.*, "KVQuant: Towards 10 million context length LLM inference with KV cache quantization," in *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024, pp. 1270–1303.

[65] Y. F. He, L. M. Zhang, W. J. Wu *et al.*, "ZipCache: Accurate and efficient KV cache quantization with salient token identification," in *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

[66] D. Song, Y. Feng, Y. Wang *et al.*, "AttnCache: Accelerating self-attention inference for LLM prefill via attention cache," *arXiv preprint arXiv:2510.25979*, 2025. [Online]. Available: https://arxiv.org/abs/2510.25979

[67] Y. Sheng, L. M. Zheng, B. R. Yuan *et al.*, "FlexGen: High-throughput generative inference of large language models with a single GPU," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, vol. 202. PMLR, 2023, pp. 31 094–31 116.

[68] W. Lee, J. Lee, J. Seo, and J. Sim, "InfiniGen: Efficient generative inference of large language models with dynamic KV cache management," in *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Santa Clara, CA, USA, 2024, pp. 155–172.

[69] J. Foerster, Y. M. Assael, N. de Freitas *et al.*, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems 29 (NeurIPS)*, 2016.

[70] L. M. Zheng, L. S. Yin, Z. Q. Xie *et al.*, "SGLang: Efficient execution of structured language model programs," in *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

[71] Q. Q. Zhang *et al.*, "SparseX: Efficient segment-level KV cache sharing for interleaved LLM serving," *arXiv preprint arXiv:2606.01751*, 2026.

[72] J. B. Yang, B. R. Hou, W. Wei *et al.*, "KVLink: Accelerating large language models via efficient KV cache reuse," in *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025, arXiv:2502.16002.

[73] J. Y. Yao, H. C. Li, Y. H. Liu *et al.*, "CacheBlend: Fast large language model serving for RAG with cached knowledge fusion," in *Proceedings of the 20th European Conference on Computer Systems (EuroSys)*, Rotterdam, The Netherlands, 2025, pp. 94–109.

[74] Y. Liu, Z. K. Luo, H. Y. Jin *et al.*, "RedKnot: Efficient long-context LLM serving with head-aware KV reuse and SegPagedAttention," *arXiv preprint arXiv:2606.06256*, 2026.

[75] N. P. Pandey, J. Kong, L. X. Hu *et al.*, "AgentKVShift: Efficient KV cache reuse for agentic memory systems," *arXiv preprint arXiv:2607.21604*, 2026.

[76] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, Y. Zhu, and L. Cui, "Congestion control for cross-datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2074–2089, 2022.

[77] S. Yue, M. Wang, Y. Yan, W. Cheng, Z. Jiang, and Z. Zhang, "RTT- or bandwidth-bound? demystifying the KV cache transfer in large language model serving," in *Proceedings of the 2nd Workshop on Networks for AI Computing (NAIC)*, 2025, pp. 5–7.

[78] T. Heo, R. Shafipour, R. Zhao, M. Golub, M. M. Kamani, R. Borkar, M. T. Chandran, P. Zardoshti, and B. D. Rouhani, "Cross-model KV cache transfer in LLM families: A closed-form linear mapping for prefill reuse," *arXiv preprint arXiv:2608.03893*, 2026. [Online]. Available: https://arxiv.org/abs/2608.03893

[79] Y. Liu, J. Wu, Y. Liu, J. Hu, M. Li, X. Chen, and W. Chen, "HYPIC: Accelerating hybrid-attention LLM serving with position-independent caching," *arXiv preprint arXiv:2607.01299*, 2026.

[80] Y. Liu, R. Qi, L. Wang, X. Wu, J. Chen, Y. Jin, J. Shao, and X. Li, "LinearKV: One cached state suffices for position-independent caching in hybrid LLMs," *arXiv preprint arXiv:2608.11231*, 2026.

[81] I. Han, P. Kacham, A. Karbasi, V. Mirrokni, and A. Zandieh, "PolarQuant: Quantizing KV caches with polar transformation," *arXiv preprint arXiv:2502.02617*, 2025.

[82] M. Wei, Q. Y. Zhao, J. Tang *et al.*, "A survey on information announcement technologies for computing power networks," *Journal on Communications*, vol. 46, no. 9, pp. 17–31, 2025.

[83] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[84] B. Peng, E. Alcaide, Q. Anthony *et al.*, "RWKV: Reinventing RNNs for the Transformer era," *arXiv preprint arXiv:2305.13048*, 2023.

[85] J. T. Gu, J. Bradbury, C. Xiong *et al.*, "Non-autoregressive neural machine translation," in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[86] X. L. Li, J. Thickstun, I. Gulrajani *et al.*, "Diffusion-LM improves controllable text generation," in *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022, arXiv:2205.14217.