Article

Structural Efficiency in AI Infrastructure: A Diagnostic Perspective on Unlocking Stranded Capacity

Gregor Herbert Wegener

Friedrichstrasse 4, 10969 Berlin, Germany; gregor.wegener@gmail.com; Tel.: +49 179 2544522

Abstract

Contemporary AI infrastructure exhibits a capital efficiency paradox: annual expenditure exceeds $400 billion, yet effective utilization rates remain in the 30–50% range. This paper presents a diagnostic system analysis identifying three structural sources of this persistent "stranded capacity"—compute resources that are paid for and powered, yet rendered inaccessible due to coordination inefficiencies rather than hardware limitations. We define and distinguish *Ghost Compute* (active cycles without state progression), *Control Incoherence* (misaligned optimization objectives across runtime layers), and *Orchestration Overhead* (non-productive consumption in agentic and control-driven systems). Drawing on publicly documented hyperscaler engineering reports, these patterns are characterized as systemic rather than incidental. Indicative recovery bounds of 5–15% throughput improvement are estimated as achievable without additional hardware procurement. A dedicated section maps these efficiency patterns to diagnostic applications within the SORT-AI structural analysis framework [14,17], providing a formal vocabulary for reasoning about coordination-induced losses across training, inference, and agentic workloads. This analysis is intentionally diagnostic, not prescriptive: it introduces a structural vocabulary for efficiency discussions without offering benchmarks, guarantees, or implementation guidance.

Keywords: AI infrastructure efficiency; stranded capacity; ghost compute; control incoherence; orchestration overhead; hyperscale systems; diagnostic analysis; SORT-AI

---

1. The Efficiency Paradox

1.1. Utilization Metrics Capture Activity, Not Delivered Work

In large-scale AI systems, commonly used utilization metrics primarily capture execution activity rather than delivered model progress. Standard monitoring tools report kernel occupancy and scheduling density, which are valuable signals for system health, but do not directly reflect how much useful work is achieved per unit of compute.

As a result, it is entirely possible for a GPU to appear fully utilized while only a subset of its theoretical throughput is converted into effective model progress. Empirical measurements indicate that systems reporting near 100% utilization frequently operate at 20–40% Model FLOPs Utilization (MFU) [1]. This gap does not imply inefficiency in isolation; rather, it highlights that activity and productivity diverge naturally as systems grow in complexity.

At organizational scale, this distinction matters. Dashboards reviewed by infrastructure and platform teams often show saturated devices, while additional optimization headroom remains distributed across data pipelines, synchronization boundaries, and runtime coordination layers. A large-scale profiling study of more than 400 production deep learning workloads conducted by Microsoft Research observed average GPU utilization around 50%, with the remaining capacity associated with pipeline structure and coordination effects rather than hardware malfunction [1]. These observations suggest that conventional metrics describe *how busy* systems are, but not yet *how efficiently* work is translated into outcomes.

1.2. Stranded Capacity as a Structural Coordination Opportunity

The observed gap between nominal capacity and delivered work is best understood as a coordination phenomenon across software layers, not as a limitation of accelerator performance. Modern AI hardware is highly capable; the challenge lies in orchestrating memory, scheduling, and execution paths such that this capability is consistently accessible to workloads.

From this perspective, a portion of deployed compute can be described as structurally stranded: paid for, powered, and operational, yet not fully reachable by workloads under prevailing orchestration assumptions. Importantly, this capacity is not lost. It is latent.

Alibaba's Aegaeon system illustrates the recoverability of such capacity. By introducing software-defined pooling and token-level scheduling across heterogeneous accelerators, Aegaeon reduced the number of GPUs required for a fixed inference workload by 82% [3]. No new hardware was introduced; instead, existing resources were made more accessible through orchestration. This result demonstrates that significant efficiency gains can emerge from architectural alignment rather than silicon replacement.

1.3. Scaling Effects and Coordination Surfaces

As AI systems scale, coordination surfaces expand. Each additional accelerator increases the number of synchronization points, scheduling decisions, and shared resources that must align for efficient execution. In synchronous training regimes, such as AllReduce-based optimization, overall throughput is naturally shaped by the slowest participating component.

At moderate scale, such effects are episodic. At extreme scale, they become a defining operational characteristic. Meta's Llama 3 training run provides a concrete example. Operating a 16,384-GPU cluster, Meta observed 419 unexpected interruptions over a 54-day training period, averaging approximately one event every three hours [2]. These events were not anomalous failures but expected outcomes of operating complex systems at unprecedented scale.

As cluster size increases further, rare events converge toward continuous presence. Under these conditions, even modest component variability can have outsized impact on end-to-end throughput. A nominal hardware interruption rate on the order of 1% can translate into substantial effective slowdown, as synchronized workloads wait for recovery or rebalancing. Framed positively, this behavior highlights the importance and potential impact of coordination-aware design, observability, and runtime alignment as first-class efficiency levers in hyperscale AI systems.

2. Taxonomy of Structural Inefficiency

Large-scale AI infrastructures exhibit recurring efficiency patterns that are independent of specific hardware vendors, frameworks, or organizational practices. To support precise technical discussion, this section introduces a small set of neutral analytical terms that describe how compute capacity can become partially inaccessible in complex, multi-layer systems.

These terms are not intended as value judgments. Instead, they provide shared vocabulary for reasoning about coordination effects that naturally emerge when highly optimized components interact at scale.

Table 1. Taxonomy of Structural Inefficiency

Term	Definition	Distinguishing Characteristic
Ghost Compute	Active compute cycles that consume power and execution resources without advancing the observable workload state	$\neq$ Idle. Ghost Compute describes execution that is powered, scheduled, and active, yet does not translate into forward progress
Stranded Capacity	Deployed and operational compute capacity that is structurally inaccessible under current coordination constraints	$\neq$ Unavailable. Stranded capacity is present and functional, but unreachable due to placement, topology, or orchestration boundaries
Control Incoherence	Emergent behavior arising from independently correct optimization objectives across multiple runtime layers	$\neq$ Bug. Each control loop behaves as designed; inefficiency arises from their interaction rather than malfunction
Orchestration Overhead	Compute and token consumption in agentic systems that does not contribute to task completion or state resolution	$\neq$ Reasoning cost. Overhead refers to redundant retrievals, abandoned plans, or unused tool invocations

These patterns are structural rather than incidental. They arise as a natural consequence of distributed architectures, deep software stacks, and autonomous control layers operating at scale. Importantly, they do not indicate misconfiguration or engineering failure. On the contrary, they tend to appear most prominently in systems that are already highly optimized and operating near practical limits.

Independent documentation of these effects across Meta, Microsoft, Google, and Alibaba—despite divergent hardware choices, software ecosystems, and operational cultures—suggests that they represent general properties of hyperscale AI systems. As such, they provide a stable conceptual anchor for discussing efficiency opportunities without reference to specific implementations or vendors.

3. Structural Sources: Training and Inference

The efficiency patterns described in this paper manifest most clearly in training and inference workloads, where scale, synchronization, and control logic intersect. In both domains, capacity limitations typically arise not from insufficient compute, but from the way coordination surfaces expand as systems grow.

3.1. Synchronization and Interconnect Effects

At cluster scale, synchronization behavior becomes a dominant factor shaping effective throughput. In tensor-parallel and data-parallel training configurations, accelerators frequently spend a significant fraction of their execution time waiting for peer synchronization rather than performing numerical computation.

This effect has been quantified in practice. Databricks reported that scaling Llama2-70B from four to eight GPUs yielded only a 0.7$\times$ latency improvement instead of the idealized 0.5$\times$ expectation, with the deviation attributable entirely to communication overhead rather than compute inefficiency [4]. The hardware continued to operate correctly; the marginal gain diminished due to coordination cost.

As clusters grow, topology constraints further shape usable capacity. Fragmentation can render otherwise available resources inaccessible to workloads requiring contiguous placement. A system may report 20% free capacity while being unable to satisfy allocation requests because the remaining accelerators are distributed across non-adjacent racks or incompatible NVLink domains. In such cases, capacity is not absent but structurally unreachable under current placement constraints [5]. Similar scheduling-induced fragmentation effects have been observed in cloud VM placement, where optimization of resource allocation itself becomes a structural challenge [11].

These effects are not anomalies. They emerge naturally in high-performance environments where accelerators, interconnects, and schedulers are optimized independently and then composed at scale.

3.2. Runtime Control Layer Conflicts

Inference and serving environments introduce an additional layer of coordination complexity through autonomous runtime controls. Schedulers and serving engines often optimize for different, locally reasonable objectives.

A cluster scheduler may observe low compute utilization and respond by injecting additional load, while the serving engine operates in a memory-bound regime due to KV-cache fragmentation or paging constraints. From the scheduler's perspective, unused compute represents opportunity; from the serving engine's perspective, memory bandwidth is already saturated. Both components behave as designed, yet their interaction can limit effective throughput.

Meta's engineering teams documented this dynamic explicitly. Inference clusters were intentionally operated at 30–50% utilization to preserve tail latency margins [6]. By refining load distribution and control interactions, they recovered 35% additional throughput on unchanged hardware. The capacity existed throughout; it became accessible once control objectives were better aligned.

Stability mechanisms exhibit similar structural trade-offs. Recovery techniques such as retries and checkpointing are essential for reliability at scale, yet they consume non-trivial resources. Meta's Llama 3 training runs devoted approximately 2.1% of total training time to checkpointing alone [2]. As cluster size increases, such overhead scales with both failure frequency and coordination scope.

Viewed structurally, these behaviors reflect the cost of operating near the practical limits of distributed execution. They highlight opportunities for improved alignment between runtime layers rather than deficiencies in individual components.

4. Structural Sources: Agentic Systems

Agentic systems represent a qualitative shift in how inference workloads are structured. Rather than executing a single forward pass, these systems orchestrate sequences of planning, tool invocation, observation, and revision. This enables powerful new capabilities, while also introducing coordination effects that differ fundamentally from traditional inference pipelines.

4.1. Scope Limitation

> *Infrastructure teams do not control the internal reasoning steps of a model (e.g., Chain-of-Thought or reasoning token generation). This analysis focuses exclusively on the orchestration environment—the loops, retrieval operations, and tool invocations that surround model execution.*

This boundary is intentional. The observations below do not address model architecture or training strategy. They concern the runtime structures in which models are embedded.

4.2. Orchestration Loops

Agentic workflows introduce recursive execution patterns that are largely absent in conventional inference. A typical Plan→Execute→Observe→Replan cycle may iterate multiple times before convergence, particularly when explicit cost-awareness is not encoded into the orchestration logic.

In well-designed systems, such iteration enables robustness and adaptability. At scale, however, these loops can consume substantial compute and token budgets without proportionally advancing task state. Industry analyses indicate that approximately 29% of enterprise AI expenditure is associated with inference-side inefficiencies in complex deployments [7]. In exploratory or unmanaged agentic environments, extreme token amplification factors have been observed, reflecting the absence of structural feedback rather than malfunctioning components.

These behaviors emerge from composition: planning modules, retrieval systems, and tools are each locally effective, yet their interaction can produce extended execution paths.

4.3. Categories of Non-Productive Token Consumption

Non-productive token consumption in agentic systems is not monolithic. It can be usefully categorized into distinct structural patterns:

1. **Ghost Tokens**: Tokens that are generated during intermediate reasoning or exploration phases but do not contribute to the final response or action.
2. **Ghost Planning**: Planning cycles that are executed and evaluated, then superseded or abandoned as the agent revises its approach.
3. **Ghost Tool-Calls**: External API or tool invocations whose results are not incorporated into subsequent decisions or are rendered obsolete by later steps.

Token efficiency studies report overhead factors of 1.5–4× in certain open agentic configurations compared to tightly optimized systems [8]. Importantly, this overhead does not scale linearly. As the number of agents, tools, and context sources increases, the space of possible execution paths expands rapidly, creating additional opportunities for redundancy.

Postmortems of early enterprise deployments illustrate this effect concretely. In poorly optimized retrieval-augmented generation pipelines, monthly costs exceeding $2,400 for fewer than 50 queries have been reported—an outcome driven by orchestration structure rather than model capability [9].

From a structural perspective, these patterns highlight the importance of visibility and feedback within agent runtimes. The compute is functioning as designed; the opportunity lies in aligning orchestration pathways more closely with task progression.

5. Recovery Opportunity: Virtual Capacity

Structural inefficiencies do not merely represent lost performance; they represent recoverable system headroom. When framed structurally rather than operationally, efficiency recovery becomes a scaling mechanism in its own right.

5.1. Capital-Efficient Scaling

Structural efficiency recovery is the most capital-efficient scaling method currently available. In an environment constrained by accelerator supply, power density, and deployment lead times, recovering 10–15% throughput from existing infrastructure is operationally equivalent to receiving additional GPU capacity—without procurement delay, incremental power draw, or capital expenditure.

This effect has been observed in practice. Meta reported a 35% throughput increase in inference workloads through structural load-balancing and tail-utilization optimization alone, operating on unchanged hardware [6]. Similar effects have been documented in retrieval-augmented systems, where eliminating redundant context loading and retrieval steps yielded substantial cost reductions without architectural replacement. Analogous capacity recovery through structural coordination has been observed in non-AI infrastructure as well, such as vertical CPU scaling approaches that reduce provisioned capacity while maintaining reliability [10].

The critical observation is that this capacity was always present. It became accessible once coordination constraints were reduced.

5.2. Indicative Recovery Bounds

Based on documented loss mechanisms across training, inference, and agentic systems, conservative indicative recovery bounds can be estimated without assuming hardware changes or runtime substitution.

Table 2. Indicative Recovery Bounds

Effect Category	Indicative Bound	Derivation
Throughput Recovery (synchronization stabilization)	5–15%	Inverse of documented synchronization-induced throughput loss
Ghost Compute Elimination	5–15%	Baseline overhead inferred from control-layer interaction analysis
Orchestration Overhead Reduction	10–25%	Token-level analysis of redundant planning, retrieval, and tool usage

Validity Conditions. These bounds assume no hardware modification, no runtime replacement, and no changes to model architecture. Recovery effects tend to amplify with system scale and are most pronounced in synchronization-intensive, multi-tenant, and agentic environments.

Evidence Qualification. The values above are indicative bounds derived from loss-pattern analysis rather than controlled benchmark measurements. Actual recovery potential depends on workload characteristics, orchestration design, and operational constraints.

Viewed collectively, these effects define a form of *virtual capacity*: performance unlocked through structural clarity rather than silicon expansion.

6. Structural Diagnostics and SORT Application Mapping

The efficiency taxonomy introduced in this paper—Ghost Compute, Control Incoherence, Orchestration Overhead, Stranded Capacity—describes observable patterns. These patterns align with structural conditions that have been independently characterized within the SORT diagnostic framework. This section maps the efficiency taxonomy to specific diagnostic applications, treating each as a structural instrument for identifying the conditions from which inefficiency arises, rather than as a remediation proposal.

The applications referenced below are drawn from the SORT-AI research series, which provides a diagnostic vocabulary for reasoning about structural instability and efficiency loss across physical, logical, and control coupling layers in AI systems [14–17]. The theoretical foundation for the operator-projection framework underlying these applications is documented in [12], with the safety-relevant projection properties described in [13].

6.1. ai.04 — Runtime Control Coherence (Cluster C)

Structural condition diagnosed: Incoherence between scheduler, orchestrator, runtime, and policy enforcement layers, producing instability and efficiency loss that is invisible at the component level but manifest at the system level [14]. The catalog definition—*diagnose and reduce incoherence between scheduler, runtime and model control loops*—addresses the core structural condition underlying both Control Incoherence and Ghost Compute: independently correct control loops whose interaction produces emergent inefficiency.

Runtime Control Coherence is the primary diagnostic for the efficiency patterns described in this paper.

Relevance to efficiency taxonomy: The scheduler-vs-serving-engine conflict documented in Section 3.2—where a scheduler injects load into a memory-bound serving environment—is a direct instance of the structural condition ai.04 diagnoses. More broadly, any regime in which runtime layers optimize for incompatible objectives represents a control coherence deficit that converts active compute into ghost cycles. The 35% throughput recovery documented by Meta [6] is consistent with the diagnostic expectation: once control objectives are aligned, capacity that was present but inaccessible becomes available.

Reference: Wegener, G.H. (2026). *SORT-AI: Runtime Control Coherence in Large-Scale AI Systems*. Preprints.org, doi:10.20944/preprints202601.0298.v1

6.2. ai.01 — Interconnect Stability Control (Cluster A)

Structural condition diagnosed: Structural stability degradation and performance collapse induced by interconnect coupling effects in distributed AI and HPC systems [15]. The catalog definition—*structural stability diagnostics and control for interconnect induced performance collapse in distributed AI and HPC systems*—addresses the physical coupling dimension from which topology-induced stranded capacity arises.

Interconnect Stability Control diagnoses conditions in which physical and logical coupling through network interconnects creates structural dependencies that go beyond bandwidth and latency metrics, producing capacity that is topologically present but operationally unreachable.

Relevance to efficiency taxonomy: The topology-induced stranded capacity described in Section 3.1—where 20% free capacity cannot satisfy allocation requests because accelerators are distributed across non-adjacent racks—is a direct instance of interconnect-induced inaccessibility. The synchronization overhead that converts scaling from a 0.5× idealized improvement to a 0.7× observed improvement [4] similarly reflects coupling effects at the interconnect layer. ai.01 diagnostics characterize these effects structurally, distinguishing between capacity that is absent and capacity that is present but unreachable under current topology and placement constraints.

Reference: Wegener, G.H. (2026). *SORT-AI: Interconnect Stability and Cost per Performance in Large-Scale AI Infrastructure*. Preprints.org, doi:10.20944/preprints202601.0161.v1

6.3. ai.13 — Agentic System Stability (Cluster D)

Structural condition diagnosed: Instability in agent workflows arising from non-linear coupling between retry loops, self-verification mechanisms, and tool-calling patterns [16]. The catalog definition—*stability control for agent workflows with retry loops, self verification, and tool calling patterns*—addresses the structural conditions under which orchestration loops generate non-productive compute consumption.

Agentic System Stability diagnostics address the efficiency taxonomy's Orchestration Overhead category.

Relevance to efficiency taxonomy: The ghost tokens, ghost planning, and ghost tool-calls categorized in Section 4.3 are instances of orchestration overhead arising from the structural properties ai.13 diagnoses. When Plan→Execute→Observe→Replan cycles iterate without cost-awareness, the resulting token amplification factors (1.5–4× documented overhead [8]) reflect non-linear coupling between planning, retrieval, and tool-invocation subsystems. ai.13 diagnostics characterize these coupling patterns structurally, identifying conditions under which orchestration loops consume resources without proportionally advancing task state.

Reference: Wegener, G.H. (2026). *SORT-AI: Agentic System Stability in Large-Scale AI Systems*. Preprints.org, doi:10.20944/preprints202601.1741.v1

6.4. ai.17 — Fault-Recovery Collapse Prevention (Cluster C)

Structural condition diagnosed: Instability introduced by recovery mechanisms themselves—checkpointing, restart, replication, and proactive migration—when these mechanisms interact with runtime state under conditions of lost coherence [17]. The catalog definition—*analysis of instability through checkpointing, restart, replication, and proactive migration mechanisms*—addresses the structural boundary at which resilience mechanisms transition from stabilizing to destabilizing behavior.

Fault-Recovery Collapse Prevention diagnostics analyze conditions under which resilience mechanisms transition from productive overhead to ghost compute.

Relevance to efficiency taxonomy: The checkpointing overhead documented in Section 3.2—2.1% of total training time for Meta's Llama 3 runs [2]—represents the baseline cost of recovery mechanisms under coherent operation. Under scaling stress, where interruption frequency increases (419 events over 54 days at 16,384-GPU scale), recovery mechanisms consume increasing shares of system capacity. ai.17 diagnostics identify the structural boundary at which recovery transitions from productive

resilience into ghost compute: cycles consumed by checkpointing, restart, and rebalancing that do not advance workload state. The diagnostic does not question the necessity of recovery mechanisms; it characterizes the conditions under which their cost profile shifts.

Reference: Wegener, G.H. (2026). *SORT-AI: Structural Efficiency Recovery in Hyperscale AI Systems*. Preprints.org, doi:10.20944/preprints202602.0015.v1

6.5. ai.27 — Inference Pipeline Control Coherence (Cluster C)

Structural condition diagnosed: Incoherence across inference pipelines including batching, caching, serving control loops, and the execution paths that connect model output to system action [14]. The catalog definition—*structural coherence analysis of inference pipelines including batching, caching, and serving control loops*—extends the ai.04 diagnostic to the specific execution paths within inference and serving environments.

Inference Pipeline Control Coherence addresses the interaction between control decisions, memory management, KV-cache behavior, and load distribution within inference pipelines.

Relevance to efficiency taxonomy: The KV-cache fragmentation and memory-bandwidth saturation effects described in Section 3.2 are instances of inference pipeline incoherence. When a scheduler optimizes for compute utilization while the serving engine is constrained by memory bandwidth, the pipeline operates under conflicting control assumptions. ai.27 diagnostics map these control paths structurally, identifying where batching decisions, cache management, and serving control interact incoherently. The diagnostic is particularly relevant to the stranded capacity pattern: inference pipelines that report available compute while being memory-bound represent capacity that is present at one layer but inaccessible at another due to control-path incoherence.

Reference: Wegener, G.H. (2026). *SORT-AI: Runtime Control Coherence in Large-Scale AI Systems*. Preprints.org, doi:10.20944/preprints202601.0298.v1

6.6. Summary of Diagnostic Mapping

Table 3. SORT Diagnostic Application Mapping — Efficiency Taxonomy

App.	Cluster	Taxonomy Mapping	Efficiency Relevance	Primary Ref.
ai.04	C	Control Incoherence, Ghost Compute	Scheduler–runtime conflicts; control objectives producing emergent inefficiency	Control [14]
ai.01	A	Stranded Capacity	Topology-induced inaccessibility; synchronization overhead at scale	Interconnect [15]
ai.13	D	Orchestration Overhead	Token amplification; ghost tokens, planning, and tool-calls in agent loops	Agentic [16]
ai.17	C	Ghost Compute	Checkpointing and restart overhead transitioning from resilience to ghost cycles	Efficiency [17]
ai.27	C	Stranded Capacity, Control Incoherence	Batching–caching–serving conflicts; memory-bound capacity invisible to compute metrics	Control [14]

These diagnostics do not claim that the efficiency losses described in this paper can be eliminated. They identify the *structural conditions* from which these losses arise, and they provide a vocabulary for reasoning about coordination-induced inefficiency at a level that is independent of specific hardware, runtime, or framework choices.

6.7. Engagement and Licensing Context

The diagnostic applications referenced in this paper are instruments within the Structural Operator Resonance Theory (SORT) framework [12]. They are designed to *diagnose*, *map*, and *characterize* structural conditions in complex technical systems. They do not constitute products, tools, or services. The methodological basis for applying these diagnostics in infrastructure risk and efficiency contexts is documented in [18].

In engagement contexts, SORT applications serve as structured instruments for:

- **Architectural review:** Identifying structural preconditions for coordination-induced efficiency loss in system designs prior to or independent of specific performance incidents.
- **Control surface mapping:** Characterizing the topology of control interactions through which runtime layers produce emergent inefficiency.
- **Pre-deployment assessment:** Surfacing structural conditions that constrain a system's efficiency envelope before operational deployment or scaling decisions.
- **Post-incident characterization:** Providing a structural vocabulary for reasoning about efficiency degradation events without attributing fault to individual components or teams.

Each application diagnoses a structural condition. It does not prescribe a remedy, guarantee an outcome, or offer a mitigation. The diagnostic output characterizes *what is structurally present* in an architecture; decisions about how to respond to that characterization remain with the system's operators and architects.

SORT applications are available through structured licensing arrangements that reflect their diagnostic scope and the domain in which they are applied. The licensing architecture is described in separate documentation and is not part of this analysis.

Scope, Non-Claims, and Intended Use

This section clarifies the analytical scope of this paper, explicitly delineates non-claims, and outlines the intended audience and use. The goal is to support constructive technical dialogue while avoiding misinterpretation of the analysis as evaluative, prescriptive, or commercial.

What This Analysis Is

This paper provides:

- A **diagnostic perspective** that introduces shared vocabulary for discussing structural efficiency in large-scale AI systems
- A **pattern characterization** derived from publicly documented hyperscaler and platform engineering observations
- An **indicative analytical framework** for identifying where coordination-related efficiency losses may emerge across training, inference, and agentic workloads

The analysis is intentionally system-agnostic. It does not assume a specific hardware vendor, runtime stack, or deployment topology.

What This Analysis Is Not

To avoid ambiguity, the following explicit non-claims apply:

Table 4. Explicit Non-Claims

Non-Claim	Explanation
Not a benchmark	No comparative performance measurements or vendor evaluations are provided
Not a guarantee	Indicative bounds are inferred from loss-pattern analysis, not from controlled experimental trials
Not a product	No tool, software component, integration, or service is proposed or offered
Not prescriptive	No implementation guidance is given; diagnosis precedes any intervention
Not model-internal	The analysis does not address LLM reasoning efficiency, Chain-of-Thought optimization, or model architecture design

These boundaries are intentional and reflect a focus on structural understanding rather than optimization execution.

Intended Audience and Use

Primary Audience. Infrastructure and Platform Engineering teams responsible for interconnects, scheduling, and fleet operations, as well as AI Runtime teams responsible for serving stacks, control layers, and orchestration behavior.

Secondary Audience. FinOps and Capacity Economics teams translating structural inefficiencies into cost and utilization signals, and Strategy or CTO offices seeking system-level framing for efficiency investment decisions.

Intended Use. This analysis is intended to serve as:

- a diagnostic lens for internal infrastructure and runtime reviews,
- a vocabulary anchor for cross-team discussions on efficiency and coordination,
- a starting point for identifying system-specific loss patterns.

It is not intended as a substitute for system-specific measurement, experimentation, or architectural decision-making.

7. Conclusion

Large-scale AI infrastructure exhibits measurable gaps between nominal capacity and delivered work. Across training, inference, and agentic workloads, publicly documented observations indicate that a non-trivial share of deployed compute remains structurally inaccessible due to coordination effects rather than hardware limitations. These effects manifest as synchronization-induced idle cycles, control-layer misalignment, and orchestration overhead that does not advance system state.

By introducing formal definitions for *Ghost Compute*, *Stranded Capacity*, *Control Incoherence*, and *Orchestration Overhead*, this analysis provides shared vocabulary for discussing efficiency phenomena that span technical and organizational boundaries. Framing these effects structurally allows observations from heterogeneous systems to be compared, categorized, and reasoned about without attributing fault to individual components or teams.

Indicative recovery bounds in the range of 5–15% throughput improvement suggest that meaningful capacity gains may be achievable through improved coordination alone, without additional hardware procurement. These bounds are not presented as guarantees, but as signals derived from documented loss mechanisms observed at scale.

The mapping of these efficiency patterns to diagnostic applications within the SORT-AI framework provides a formal vocabulary for reasoning about coordination-induced losses at a structural level. The diagnostics identify conditions from which inefficiency arises—control incoherence, interconnect-induced inaccessibility, recovery amplification, orchestration coupling—without prescribing specific interventions.

This paper is intentionally diagnostic rather than prescriptive. It does not claim that existing engineering approaches are insufficient, nor does it assert that operators lack awareness of these dynamics. Instead, it offers a structured perspective for organizing complex system behavior into

analytically tractable categories—supporting informed decision-making without dictating specific interventions.

In large systems, improved observability of structural behavior is often the first step toward more predictable, efficient operation.

Acknowledgments: The author acknowledges prior internal architectural work that informed the conceptual development of the diagnostic perspective presented in this paper. The author also acknowledges prior work within the SORT framework that informed the structural diagnostic mapping.

Conflicts of Interest: The author declares no conflicts of interest. The author is the developer of the SORT framework referenced in the diagnostic mapping section.

Data Availability Statement: No new data were generated in this study. All referenced data are available in the cited publications and publicly accessible sources listed in the references.

1. Jeon, M.; Venkataraman, S.; Phanishayee, A.; et al. (2024). Characterization and Prediction of Deep Learning Workloads in Large-Scale GPU Datacenters. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '24)*.
2. Dubey, A.; Jauhri, A.; Pandey, A.; et al. (2024). The Llama 3 Herd of Models. *arXiv preprint* arXiv:2407.21783.
3. Zhai, E.; et al. (2025). Aegaeon: Effective GPU Pooling for Concurrent LLM Serving on the Market. *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP '25)*.
4. Databricks Engineering. (2024). LLM Inference Performance Engineering: Best Practices. *Databricks Engineering Blog*. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices
5. Tirmazi, M.; et al. (2020). Borg: The Next Generation. *Proceedings of the 15th European Conference on Computer Systems (EuroSys '20)*, 1–14.
6. Meta Engineering. (2024). Taming Tail Utilization of Ads Inference at Meta Scale. *Meta Engineering Blog*. https://engineering.fb.com/2024/07/10/production-engineering/tail-utilization-ads-inference-meta/
7. IDC & DataRobot. (2025). The Hidden AI Tax: Cost Control in the Age of GenAI and Agentic Workflows. *IDC Market Spotlight*.
8. Nous Research. (2025). Token Efficiency Across Language Models. *Technical Report*.
9. Stevens Institute of Technology. (2025). The Hidden Economics of AI Agents: Managing Token Costs and Latency Trade-offs. *Stevens Online Blog*. https://online.stevens.edu/blog/hidden-economics-ai-agents-token-costs-latency/
10. Uber Engineering. (2024). Vertical CPU Scaling: Reduce Cost of Capacity and Increase Reliability. *Uber Engineering Blog*. https://www.uber.com/blog/vertical-cpu-scaling/
11. Google Research. (2024). Solving Virtual Machine Puzzles: How AI is Optimizing Cloud Computing. *Google Research Blog*.
12. Wegener, G.H. (2025). The Supra-Omega Resonance Theory (SORT): A Closed Structural Architecture for Cross-Domain Scientific Analysis. *Preprints.org*, 2025. doi:10.20944/preprints202511.1783.v3
13. Wegener, G.H. (2025). SORT-AI: A Projection-Based Structural Framework for AI Safety—Alignment Stability, Drift Detection, and Scalable Oversight. *Preprints.org*, 2025. doi:10.20944/preprints202512.1334.v2
14. Wegener, G.H. (2026). SORT-AI: Runtime Control Coherence in Large-Scale AI Systems—Structural Causes of Cost, Instability, and Non-Determinism Beyond Interconnect Failures. *Preprints.org*, 2026. doi:10.20944/preprints202601.0298.v1
15. Wegener, G.H. (2026). SORT-AI: Interconnect Stability and Cost per Performance in Large-Scale AI Systems—Structural Causes, Diagnostic Operators, and Infrastructure-Level Consequences. *Preprints.org*, 2026. doi:10.20944/preprints202601.0161.v1
16. Wegener, G.H. (2026). SORT-AI: Agentic System Stability in Large-Scale AI Systems—Structural Causes of Cost, Instability, and Non-Determinism in Multi-Agent and Tool-Using Workflows. *Preprints.org*, 2026. doi:10.20944/preprints202601.1741.v1
17. Wegener, G.H. (2026). SORT-AI: Structural Efficiency Recovery in Hyperscale AI Systems—A Diagnostic Framework for Throughput, Control, and Orchestration Losses. *Preprints.org*, 2026. doi:10.20944/preprints202602.0015.v1

18. Wegener, G.H. (2026). An Operator-Projection Framework for Structural Risk Assessment in AI Infrastructure: Architecture, Applications, and Evidence Requirements. *SSRN Electronic Journal*, 2026. https://doi.org/10.2139/ssrn.6094907