Extending the Sovereign AI OS: Symphony as Compute Ontology for Palantir Foundry and NVIDIA

Technical Paper
March 30, 2026

Kevin D. Johnson, MBA, MAcc, MSGTD, MAT

DOI: 10.5281/zenodo.19889988

***Abstract*—The Sovereign AI Operating System Reference Architecture, announced by Palantir and NVIDIA in March 2026, combines Foundry's data ontology with NVIDIA's GPU stack in a turnkey AI datacenter blueprint. The present paper extends the architecture with IBM Spectrum Symphony as a heterogeneous compute orchestrator, enabling neuromorphic processors, quantum resources, edge sensors, and mainframe systems to participate as peer compute tiers alongside GPUs within Foundry's governed ontology framework. Four working demonstrations validate the extension: autonomous ontology construction through multi-domain neuromorphic perception spanning seven sensor modalities, 155:1 data distillation from edge events to Foundry records through cross-modal neuromorphic fusion, computational diversity as a security property through multi-paradigm trust verification, and AI-enabled financial ontology discovery. Symphony's compute ontology, comprising typed ELIM resource metrics, consumer hierarchy governance, and SOAM service lifecycle management, provides workload management intelligence the existing compute substrate cannot express. The architecture is complementary to the existing reference architecture rather than competitive, adding a heterogeneous orchestration layer while maintaining all existing Palantir and NVIDIA components.**

I. Introduction

Palantir Technologies and NVIDIA announced the Sovereign AI Operating System Reference Architecture (AIOS-RA) on March 12, 2026, defining a turnkey AI datacenter blueprint spanning hardware procurement through application deployment [1]. The architecture combines NVIDIA Blackwell Ultra GPUs with Spectrum-X networking, NVIDIA AI Enterprise software, and Nemotron open models alongside Palantir's Foundry, AIP (Artificial Intelligence Platform), and Apollo deployment infrastructure. The reference architecture targets organizations with data sovereignty requirements, latency-sensitive workflows, and existing GPU infrastructure. Jensen Huang characterized the partnership as "a next-generation engine to fuel AI-specialized applications and agents" [1].

The AIOS-RA makes a significant contribution by integrating data intelligence (Palantir Foundry's ontology) with accelerated computing (NVIDIA's GPU stack) in a single reference architecture. The architecture correctly identifies that AI deployment requires both computational power and structured operational knowledge, addressing the gap between raw model capability and operational decision-making. The ontology layer provides the semantic structure and the GPU layer provides the computational substrate. The combination enables AI agents operating within a governed ontology rather than against unstructured data.

The compute orchestration layer of the AIOS-RA, however, inherits a structural limitation. The architecture uses hardened Kubernetes as its compute substrate, with Palantir's Rubix providing zero-trust security and NVIDIA's Run:ai providing GPU scheduling within the Kubernetes framework [1]. Kubernetes treats compute resources as containers requesting scalar quantities of CPU, memory, and GPU count. The scheduling model cannot natively express per-resource-type operational metrics, heterogeneous temporal cadences across compute paradigms, or service graph composition across resources differing in kind. The architecture commits what prior work on heterogeneous orchestration has termed the homogenization error: achieving unity across the compute domain by erasing the operational distinctions among different resource types [2].

The present paper proposes extending the AIOS-RA with IBM Spectrum Symphony as the heterogeneous compute orchestrator, complementing rather than replacing the existing Palantir and NVIDIA components. Symphony's Service-Oriented Application Manager (SOAM) and External Load Information Manager (ELIM) provide the per-resource-type metric reporting, service graph composition, and heterogeneous temporal orchestration the Kubernetes substrate cannot express [3], [4]. The extension enables compute tiers the AIOS-RA does not contemplate, including neuromorphic processors, quantum resources, edge sensors, and mainframe systems to participate as peer resource types alongside GPUs within the ontology-driven operational framework Foundry provides.

Palantir's existing edge capability, the Sensor Inference Platform (SIP), deploys models to edge devices through Apollo's CI/CD pipeline and collects telemetry for retraining [5]. SIP addresses software delivery to edge environments but does not

provide real-time workload orchestration across heterogeneous compute resources with per-resource operational metrics. The gap between deploying a model to a sensor and orchestrating a fleet of sensors, GPUs, neuromorphic chips, and quantum resources in real time under unified scheduling is the gap the present paper addresses.

The paper grounds the proposal in four working demonstrations integrating Palantir Foundry, NVIDIA GPUs, BrainChip AKD1000 neuromorphic processors, and IBM Spectrum Symphony. SymWisdom demonstrates autonomous ontology construction through multi-domain neuromorphic perception spanning seven sensor modalities, from L-band satellite RF through visible light through acoustic, with Nemotron-3-Super-120B crystallizing discovered concepts as Foundry ontology objects [6]. The SymPalantir-Akida sensor fusion system fuses the same seven modalities across ten neuromorphic chips with cross-modal emergence logic, achieving a 155:1 data distillation ratio from edge events to Foundry records [7]. SymPalantir RAG demonstrates heterogeneous compute diversity as a security property, with neuromorphic anomaly detection, GPU-based semantic review, and homomorphic encryption providing independent trust decisions across fundamentally different computational paradigms [8]. SymCognitive demonstrates AI-enabled ontology creation for financial services, discovering 1,804 entities and 2,690 relationships from financial data through Foundry's Ontology Intelligence [9].

The contributions of the paper are fivefold. First, the paper identifies a specific architectural limitation in the AIOS-RA's Kubernetes-based compute substrate and proposes Symphony as the heterogeneous orchestration layer addressing the limitation. Second, the paper demonstrates working integrations of Palantir Foundry, NVIDIA GPUs, and BrainChip neuromorphic processors under Symphony orchestration, extending the AIOS-RA's compute model beyond GPU-only infrastructure. Third, the paper presents autonomous ontology construction as a novel capability enabled by heterogeneous compute, in which neuromorphic perception at the edge feeds LLM reflection feeding Foundry ontology growth without human schema design. Fourth, the paper demonstrates through a targeted test scenario that computational diversity across fundamentally different hardware architectures can produce security properties no single-paradigm architecture achieves, with broader adversarial testing identified as future work. Fifth, the paper positions the extended architecture as a multi-cloud and hybrid deployment model, leveraging Palantir's cloud-native deployment via Apollo alongside Symphony's on-premise and cloud scheduling capabilities.

II. Two Ontologies: Data Intelligence and Compute Intelligence

A. Palantir Foundry's Data Ontology

Palantir Foundry's ontology is not a database schema. The ontology is an operational layer mapping digital assets to their real-world counterparts, modeling operational processes as object types (nouns) and action types (verbs) connected by link types (relationships) [10]. The ontology mediates between raw data and operational decisions. A dataset of GPS coordinates becomes a fleet tracking system when the ontology defines Vehicle objects, Route links, and PositionUpdate actions. A dataset of financial transactions becomes a risk monitoring system when the ontology defines Account objects, TransactionFlow links, and FraudAlert actions.

The Ontology Metadata Service (OMS) defines all ontological entities. Object Storage V2 supports tens of billions of objects per type with incremental indexing. The Object Set Service (OSS) handles read operations including searching, filtering, and aggregating. The Actions Service provides structured edits with permissions and conditions. AIP operates within the ontology, with agents accessing objects, triggering actions and reading relationships through governed interfaces rather than querying raw data [10], [11].

The ontology's operational power depends on the compute infrastructure feeding it. An ontology populated by a single compute paradigm reflects the perceptual capabilities of that paradigm alone. A GPU-only compute substrate feeds the ontology with LLM-generated classifications, computer vision detections, and recommendation outputs. The ontology captures what GPUs can perceive. A heterogeneous compute substrate encompassing neuromorphic perception, quantum-enhanced features, mainframe settlement records, and edge sensor observations feeds the ontology with a richer and more diverse operational picture. The ontology can then capture what the full spectrum of computational paradigms can perceive.

B. Symphony's Compute Ontology

Most readers will be familiar with Symphony as an HPC workload manager. What is less widely known is that Symphony's resource model is itself a formal ontology of the compute domain, a dynamic compute platform parallel in structure to Foundry's data ontology. The resource model specifies what entities exist, what properties those entities have, what relationships govern them, and what operations apply to them. The structural parallel is precise and deliberate.

Foundry's ontology operates across three layers: the Ontology Metadata Service defines what object types exist and what properties they carry; link types define relationships between objects; action types define what operations apply. Symphony's compute ontology operates across three corresponding layers with the same structural logic.

Resource schema (parallel to Foundry's object types). Symphony's `ego.shared` Resource block is a type system for the compute domain. Each resource declaration specifies a name, a data type (Boolean, Numeric, String), a reporting interval, and a semantic direction indicating whether higher values represent more capacity or more load. When an administrator declares `akida_device_ready Boolean` and `akida_power_mw Numeric 10 N`, the administrator creates a typed entity in the compute ontology: a neuromorphic chip exists in the domain, its availability is Boolean, its power consumption is numeric and reported every 10 seconds and lower power consumption means more available capacity.

The declaration `vllm_kv_cache_usage_pct Numeric 10 Y` creates a different entity with different semantics: an inference cache exists, its fill level is numeric and higher values mean less available capacity. ELIM scripts populate the schema with real-time values from every host and scheduling expressions operate over the typed metrics. A `resReq` expression such as `select(akida_device_ready && gpu_util < 80)` is a query over the compute ontology, selecting resources whose typed properties satisfy the expression. The production deployment described in the present paper reports over 100 unique resource metrics across neuromorphic, GPU, vLLM inference, sensor, drone telemetry, and standard system categories.

Consumer hierarchy (parallel to Foundry's link types and governance). Foundry's link types define how objects relate to each other and Foundry's security model governs who can access what objects under what conditions. Symphony's consumer hierarchy defines how workload tenants relate to each other and how resources flow between them. The hierarchy is a tree structure operating across multiple governance dimensions simultaneously.

Prioritization and preemption. Each consumer carries a configurable priority level. When demand exceeds supply the scheduler preempts lower-priority consumers to restore guaranteed shares to higher-priority consumers. Preemption policies are per-consumer and per-resource-group, enabling fine-grained control over which workloads yield under contention.

Lending and borrowing. Idle capacity flows laterally to siblings and vertically through the tree. When `EGO_RECLAIM_FROM_SIBLINGS=Y` the scheduler automatically lends unused capacity from idle consumers to active siblings and reclaims the capacity when the owner's demand returns. The rebalancing cycle operates every second (`EGO_DISTRIBUTION_INTERVAL=1`), ensuring lending and reclamation occur at scheduling-relevant timescales. A consumer owning 5% of a resource group can use 100% if no siblings have demand and yield instantly when siblings activate.

Multi-dimensional resource planning. Each consumer holds independent resource policies per resource group. The consumer `/PortfolioServices` holds policies across five resource groups simultaneously (ManagementHosts, ComputeHosts, vllm_gpu_rg, akida_rg, pqfm_gpu_rg), with different share percentages, planned quotas, and share quotas per group. The same physical GPU host participates in three resource groups (gpu_rg, vllm_gpu_rg, and pqfm_gpu_rg) with different slot counts, meaning the same hardware serves general GPU workloads, vLLM inference, and quantum computation under different governance policies. Resource planning is inherently multi-dimensional: a consumer's entitlement is not a single number but a vector across all resource groups the consumer participates in.

Organizational modeling. The consumer hierarchy typically mirrors organizational structure, and the flexibility of the tree model accommodates multiple organizational patterns. A line-of-business design places organizational units (trading desk, risk analytics, and settlement) as top-level consumers with applications as children. An application-centric design places applications (pricing engine, risk engine, and reporting) as top-level consumers with organizational cost centers as children. A hybrid design nests both. The same cluster can reorganize its consumer hierarchy without redeploying services because the governance model is independent of the workload model. Financial institutions routinely restructure consumer hierarchies to reflect organizational changes, regulatory reporting requirements, or cost allocation models.

Production Symphony deployments at major financial institutions manage thousands of consumers across deep hierarchy levels, processing billions of tasks daily for risk analytics, pricing, and settlement [3], [4]. The consumer hierarchy scales from the present demonstration cluster to institutional grids without architectural modification. The four child consumers under `/PortfolioServices` (PQFM-Compute, AkidaClassifier, VLLMNarrative, ZOSSettlement) illustrate how four entirely different computational paradigms share resources within one governance subtree.

SOAM service lifecycle (parallel to Foundry's action types). Foundry's action types define what operations apply to ontology objects, with permissions and validation conditions. SOAM defines what operations apply to compute services through a multi-dimensional lifecycle model whose expressiveness has no analogue in container orchestration.

The lifecycle operates across seven governed phases (Register, CreateService, SessionEnter, SessionUpdate, Invoke, SessionLeave, DestroyService). Each phase carries independent policies for five failure conditions (timeout, exit, return, exception-failure, exception-fatal) with orthogonal actions on the service instance (blockHost, restartService, keepAlive) and on the workload (retry, succeed, fail). A service can specify that a CreateService timeout restarts the instance while an Invoke exception retries the task while keeping the instance alive. The two failure domains are independent: the fate of the service instance and the fate of the workload are governed by separate policy axes rather than by a single pod restart policy.

Session types express distributed continuity guarantees no container lifecycle can replicate. A recoverable session with `abortSessionIfClientDisconnect=false` continues executing tasks after the client disconnects, preserving results for later retrieval. A detachable session with extended grace periods (30 seconds versus the 100 millisecond default) allows jobs to complete and detach before the framework reclaims resources. Historical persistence (`persistTaskHistory=all`) journals every task outcome for audit trails. Minimum service guarantees (`minServices=1`) ensure at least one service instance remains available during rolling updates, preventing quorum loss during version transitions.

Proactive resource governance adds a fourth dimension. Boundary management defines four escalation levels (PROACTIVE, SEVERE, CRITICAL, HALT) for memory and virtual

address space. When available memory drops to 50% the framework preempts low-priority tasks. At 40% it escalates constraints. At 25% it suspends non-essential work. At 15% it halts to prevent kernel OOM termination. Kubernetes has hard resource limits; Symphony has state-aware escalation with configurable thresholds per application.

Multi-tier recursive task execution enables parent services to spawn child tasks across the same or different hosts within a unified session context, with per-tier failure isolation and shared state through CommonData paging pools. Multi-SSM failover provides transparent task-level redirection between physical service managers when one fails, routing tasks within a live session rather than rescheduling pods after a crash. Data-aware scheduling places tasks on hosts where input data resides, avoiding network transfer latency. Dataset dependencies declare that a session requires specific datasets to be present before execution begins.

Service deployment and live update. Symphony provides its own deployment lifecycle independent of any external CI/CD system. The `soamdeploy` command distributes versioned service packages (tar.gz archives containing binaries, models, configuration, and wrapper scripts) to compute hosts through the consumer hierarchy. The `soamreg` command registers or re-registers service definitions against a running cluster without restarting EGO or disrupting other services. The `soamcontrol` command disables, enables and unblocks services individually, allowing operators to cycle a single application while the remaining 46 continue processing. Combined with `minServices=1` guarantees, the workflow supports rolling updates where at least one service instance remains available throughout the transition. Version increments in package names prevent stale cached artifacts on target hosts. The entire cycle from disable through redeploy to enable completes within minutes on the production cluster described in the present paper, with no orchestrator restart and no disruption to unrelated services. Container orchestration achieves similar outcomes through rolling deployment strategies, but Symphony's mechanism operates at the service level rather than the pod level, preserving session state and consumer governance across the update boundary.

These capabilities are not experimental extensions but production features documented in the SDK samples shipped with every Symphony release. The 99 XML application profiles included in the Symphony 7.3.2 distribution define the complete lifecycle, failure and governance semantics across 64 sample applications in eight languages (C++, Java, Python, R, .NET, MATLAB, cross-language, and generic command execution). The production deployment described in the present paper manages 47 SOAM applications, each with its own consumer binding, session types, failure policies, boundary configurations, and scheduling affinity, spanning neuromorphic inference, LLM serving, quantum circuits, mainframe settlement, and sensor processing.

The parallel between Foundry and Symphony is not a metaphor. The two systems solve the same structural problem in different domains. Foundry answers: what data entities exist, what types do they have, how do they relate, and who governs them. Symphony answers: what compute resources exist, what types do they have, how do they relate, and who governs them. The AIOS-RA provides the first ontology but not the second. The extended architecture proposed in the present paper provides both, creating unified ontological coverage across the data domain (Foundry) and the compute domain (Symphony) within one operational framework.

C. The Kubernetes Gap

The AIOS-RA's compute substrate is hardened Kubernetes with Rubix providing zero-trust security (enforced encryption, ephemeral 48-hour node cycling, FedRAMP/IL5/IL6 compliance) and Run:ai providing GPU scheduling within the Kubernetes framework [1]. Apollo manages software deployment lifecycle through pull-based release channels, health-gated promotion, and automated rollback, operating at the application delivery layer above the infrastructure layer [5]. Palantir's Resource Management provides FIFO resource queues gating concurrent vCPU and vGPU usage per project.

Kubernetes does not provide a compute ontology. Kubernetes is a container placement engine whose resource model is a flat vector of scalar quantities: CPU millicores, memory bytes and integer device counts. The gap between Kubernetes and Symphony is not incremental but categorical, comparable to the gap between a file system and a relational database. Both store data but the relational database models the types, relationships, and constraints governing the data while the file system manages bytes on disk. Kubernetes manages container placement. Symphony models the types, relationships and governance of the compute domain.

The specific limitations follow from the categorical difference. Kubernetes cannot natively express per-resource-type operational metrics with semantic direction. Kubernetes cannot natively express heterogeneous temporal cadences spanning five orders of magnitude (622 microseconds for neuromorphic classification through minutes for QPU queue time). Kubernetes cannot compose service graphs across compute paradigms at the scheduling layer. Kubernetes provides uniform pod restart policies rather than per-paradigm failure semantics. Run:ai adds GPU-count-level scheduling within Kubernetes but supports only NVIDIA GPUs, cannot manage non-containerized workloads, and has no runtime GPU enforcement preventing containers from exceeding allocated limits (KAI Scheduler Issue #423) [12]. None of these limitations is a missing feature awaiting implementation. Each is a consequence of Kubernetes' architectural position as a container placement engine rather than a compute ontology.

D. The Dual-Ontology Architecture

The extended architecture proposed in the present paper combines both ontologies under a unified operational framework. Foundry governs what the organization knows. Symphony governs how the organization computes. The combination is not additive but multiplicative: Symphony's compute

ontology enables Foundry's data ontology to receive intelligence from compute paradigms the Kubernetes substrate cannot express and Foundry's data ontology provides the semantic structure giving meaning to what Symphony's heterogeneous compute produces.

The proposal is not to replace Kubernetes in the AIOS-RA but to complement it with a compute ontology layer it lacks. Kubernetes manages container lifecycle, networking, service discovery, and security through Rubix. Symphony manages heterogeneous compute orchestration, per-resource-type scheduling and cross-paradigm workflow composition. The two operate at different layers: Kubernetes provides the infrastructure fabric; Symphony provides the compute intelligence. Each system manages its own deployment independently: Apollo deploys Palantir and Kubernetes services through pull-based release channels, and Symphony deploys its own services through soamdeploy.

The complementary model mirrors established practice in financial services, where Symphony operates alongside Kubernetes (often via OpenShift) in production environments. Large financial institutions run Symphony for latency-sensitive compute workloads (pricing, risk, and Monte Carlo simulation) and Kubernetes for microservices (APIs, dashboards, data pipelines). The two systems coexist on shared infrastructure, each managing the domain for which it was designed [3], [4].

III. Related Work

A. Palantir-NVIDIA Sovereign AI OS Reference Architecture

The AIOS-RA, announced March 12, 2026, defines a complete AI datacenter stack [1]. The hardware layer specifies NVIDIA Blackwell Ultra GPUs (8 per unit) with Spectrum-X Ethernet networking. The NVIDIA software layer includes AI Enterprise, CUDA-X libraries, Nemotron open models, and Magnum IO performance acceleration. The Palantir software layer includes Foundry (with Catalog, Build, Multipass), AIP, Apollo, and Rubix. The infrastructure foundation is hardened Kubernetes.

The architecture targets organizations with data sovereignty requirements, geographic distribution, and existing GPU infrastructure. The "sovereign" designation indicates that all data processing occurs within the customer's infrastructure perimeter, addressing regulatory and security requirements that prevent cloud-hosted AI processing. Palantir's Apollo manages autonomous deployment and lifecycle management across on-premise, cloud and air-gapped environments [5].

The present paper shares the AIOS-RA's vision of integrated data intelligence and accelerated computing but extends the compute layer to accommodate resources beyond GPUs. The extension is architecturally conservative, adding an orchestration layer above Kubernetes rather than replacing the Kubernetes substrate.

B. Palantir Foundry Architecture

Foundry's architecture comprises several backend services managing the ontology lifecycle [10]. The Ontology Metadata Service defines object types, link types, and action types. Object Storage V2 provides next-generation indexing supporting tens of billions of objects per type with up to 2,000 properties per object type. The Object Set Service handles read operations. The Object Data Funnel orchestrates writes from datasources and user edits. The Actions Service manages structured modifications with permissions and validation conditions. Functions on Objects provide operational logic for dashboards and decision-making applications.

AIP integrates commercial and open-source LLMs with the ontology through secure interfaces [11]. AIP Agent Studio creates agents operating within the governed ontology. Context engineering supports batch, streaming, and real-time CDC (change data capture) data integration through Spark, Flink, DataFusion, and Polaris runtimes. Security controls span role-based, marking-based, and purpose-based access across all human and agent operations. Full observability logs every action by humans or agents, including token consumption tracking.

The Ontology MCP (Model Context Protocol), available in beta for Developer Console users since January 2026, exposes ontology operations through the Model Context Protocol standard [10]. The present work uses Foundry's REST APIs for ontology operations; future integration through Ontology MCP would provide a standardized interface for Symphony services interacting with Foundry objects.

C. NVIDIA Nemotron-3-Super-120B

Nemotron-3-Super-120B, highlighted by Jensen Huang at GTC 2026, employs 120 billion total parameters with 12 billion active per token through a mixture-of-experts architecture [13]. The hybrid Mamba-Transformer backbone combines Mamba-2 layers for linear-time sequence processing with transformer attention layers for maintaining KV state. The architecture achieves a memory footprint of approximately 60 GB at NVFP4 precision, fitting within a single GPU's high-bandwidth memory.

The present work uses Nemotron-3-Super-120B as the reflection engine in the SymWisdom demonstration under Symphony orchestration. The model's mixture-of-experts efficiency aligns with the service-oriented inference pattern Symphony's SOAM framework manages and the model's open-source availability under NVIDIA licensing enables on-premise deployment within sovereignty-constrained environments.

D. Heterogeneous Orchestration and the Zhou Lineage

The quantum-centric heterogeneous orchestration paper demonstrated Symphony scheduling across QPU, NPU, GPU, CPU, and mainframe tiers under a single scheduling domain, with the implementation lineage tracing to Songnian Zhou's 1987 doctoral research on dynamic load balancing at UC Berkeley [2], [14]. Zhou's dissertation classified heterogeneous systems into two types: Type A systems with hosts of varying capacities but a uniform software interface, where the load index needs only rescaling and Type B systems with hosts of different architectures requiring "different, but functionally

compatible programs" on each host type [14]. The present paper operates entirely in Zhou's Type B category, coordinating resources whose computational paradigms share no common abstraction. Symphony's ELIM is the direct descendant of Zhou's load index, generalized from per-host CPU metrics to per-type arbitrary metrics including GPU utilization, neuromorphic spike rates, QPU fidelity, and inference latency [3], [4], [14]. ELIM's typed resource definitions (Boolean, Numeric, String with semantic direction and update cadence) are the engineering answer to Zhou's Type B problem, enabling the scheduler to accommodate resources differing in kind without reducing them to a common model.

The unity-in-distinction principle governing heterogeneous orchestration, holding different computational natures in unity without collapsing their differences, applies directly to the ontology-orchestration integration the present paper proposes. The ontology layer (Foundry) provides semantic unity across all data types. The orchestration layer (Symphony) provides scheduling unity across all compute types. Neither layer requires the resources or data within its domain to conform to a common abstraction. The ontology accommodates object types as different as Vehicle, Account, AcousticEvent, and WisdomCrystal. The orchestrator accommodates resource types as different as GPU, NPU, QPU, and mainframe. The combination provides both semantic and computational unity-in-distinction.

E. Palantir Edge AI and the Connected Edge

Palantir's edge capabilities are more substantial than a cursory review suggests and deserve careful characterization. Three components constitute Palantir's edge architecture.

Sensor Inference Platform (SIP) provides edge AI orchestration and sensor fusion for disconnected and resource-constrained environments [5]. SIP deploys Micro Models, modular operation-specific algorithms, to drones, aircraft, ships, robots, satellites, and IoT sensors. SIP supports model chaining across devices, ensemble outputs from multiple models and hot-swapping models at runtime without breaking sensor data flow. In the L3Harris integration, SIP ran multiple models in parallel for target detection on WESCAM MX-20 EO/IR systems, achieving 20x bitrate reduction through metadata-only downlinks [15]. SIP hardware targets include NVIDIA Jetson, Qualcomm Dragonwing processors, Anduril Menace tactical systems and industrial PCs.

Embedded Ontology provides a lightweight, edge-native instantiation of Foundry's ontology that operates independently at each site [5]. OSDK APIs provide consistent interfaces for querying assets and triggering write-back to PLCs and SCADA systems. Local CRUD operations execute with millisecond latency independent of cloud connectivity. Object Peering synchronizes ontology state bidirectionally between edge and cloud, with edge devices buffering changes during disconnection and reconciling state when connectivity restores. The Embedded Ontology runs on Single Node OpenShift (SNO) with an immutable OS where updates are atomic transactions. Apollo manages the deployment lifecycle with DISA IL5/IL6 authorization for classified environments.

Connected Edge packages compute, storage, and GPU acceleration into form factors scaling from industrial PCs to 3U rack-mounted servers, with connectivity options including wired, wireless, and 5G cellular [5]. Pre-configured protocol adapters support OPC UA and MQTT for industrial environments. GigE Vision support enables high-resolution camera integration.

These capabilities are genuinely strong for application delivery to edge environments and for maintaining ontology state at the edge. However, three architectural characteristics distinguish the edge orchestration the present paper proposes from Palantir's edge architecture.

First, Palantir's edge orchestration operates at the software deployment layer. Apollo deploys models and application updates to edge devices through canary rollouts with health-gated promotion. Symphony's SOAM orchestrates workloads across edge devices in real time through per-resource ELIM metrics. The distinction is between "what software version should run on each device" (a CI/CD problem Apollo solves) and "which device should process this inference task given its current power utilization, spike rate, and classification confidence" (a workload management problem ELIM solves).

Second, Palantir's Object Peering is eventually consistent, designed for intermittent connectivity scenarios. Symphony's IBM Storage Scale (GPFS)-mediated shared perceptual state is simultaneously readable and writable by all nodes at sensor rate with sub-millisecond latency. The distinction matters for cross-modal fusion, where observations from multiple sensor modalities must correlate within a defined spatiotemporal window. Eventually-consistent sync introduces latency incompatible with microsecond-scale neuromorphic classification.

Third, Palantir's edge architecture does not include neuromorphic hardware, spiking neural network inference, or the physics-based data distillation neuromorphic silicon provides. SIP runs conventional neural network models (ONNX runtime) on conventional hardware. The 155:1 distillation ratio the present work achieves arises from neuromorphic sparsity at the silicon level, not from software-based filtering. No conventional inference runtime on conventional hardware replicates the combination of 622-microsecond latency, 30-milliwatt power consumption, and sparsity-driven noise rejection the AKD1000 provides.

The extended architecture proposed in the present paper uses both systems, each managing its own deployment independently. Apollo deploys Palantir software (Foundry, AIP, SIP) to edge devices through its classification-authorized pipeline. Symphony deploys its own service packages through soamdeploy and orchestrates workloads in real time through ELIM-reported metrics. Foundry receives distilled intelligence through the observation-to-ontology pipeline. The systems meet at the data boundary, not the deployment layer.

F. Palantir Orchestrator and Knowledge Nodes

Two additional Palantir capabilities deserve acknowledgement because they occupy adjacent territory to capabilities the present paper claims for Symphony.

Palantir's Orchestrator provides long-running, durable execution with checkpointing, step-level retry on failures, await conditions suspending execution until ontological state changes, and agent-agent teaming through ontology writes [11]. The Orchestrator is genuinely superior to any HPC scheduler for enterprise business workflows, including document approval chains, multi-step pre-authorization, and human-in-the-loop decision processes. The present paper does not propose Symphony as a replacement for Orchestrator. Symphony schedules compute workloads across heterogeneous hardware; Orchestrator orchestrates business processes within the ontology. The two operate at different layers and are complementary.

Knowledge Nodes, introduced at DevCon 4, provide autonomous capture of procedural and episodic memory through agents operating within the ontology. Agents discover facts, procedures, and insights from data and conversation, storing them as knowledge artifacts that feed back into subsequent reasoning. Knowledge Nodes are architecturally adjacent to the SymWisdom autonomous ontology construction the present paper demonstrates. The distinction is that Knowledge Nodes capture knowledge within an existing ontology schema while SymWisdom generates new ontology object types from discovered patterns. Knowledge Nodes refine the ontology's content; SymWisdom extends the ontology's structure. Both capabilities are valuable and a combined deployment would benefit from both: Knowledge Nodes enriching objects within Foundry's governed schema and SymWisdom proposing new object types emerging from neuromorphic perception.

G. Neuromorphic Computing in Operational Systems

BrainChip's AKD1000 provides commercial neuromorphic silicon with a CNN-to-SNN conversion pipeline enabling deployment of trained convolutional networks as spiking neural networks [16]. The AKD1000 classifies at 622 microseconds latency and 30 milliwatt power consumption, operating at substantially lower latency and power consumption than GPU-based classifiers, with power ratios ranging from 67:1 against dedicated edge inference hardware to over 1,600:1 against server-class GPUs [17], [18], [19]. No prior known work has integrated neuromorphic processors with Palantir Foundry's ontology under unified compute orchestration. The present paper demonstrates neuromorphic perception feeding ontology construction through an intermediate LLM reflection layer, a pipeline no GPU-only architecture can replicate at equivalent power efficiency and latency.

IV. Architecture

A. Extended AIOS-RA Overview

The extended architecture adds Symphony as the heterogeneous compute orchestration layer to the AIOS-RA stack, maintaining all existing Palantir and NVIDIA components while enabling additional compute tiers.

B. Symphony-Foundry Integration Model

The integration between Symphony and Foundry operates through three pathways. The first pathway is demonstrated in the present work. The second and third pathways are architectural projections enabled by the demonstrated integration but not yet implemented as end-to-end systems.

Observation-to-ontology pipeline. Compute services orchestrated by Symphony produce observations, classifications, detections, and analyses. The pipeline transforms raw compute output into structured ontology objects ingested by Foundry through REST API or SDK. The transformation is a SOAM service within the same service graph as the compute service, ensuring the ontology receives data at the cadence the compute tier produces it without intermediate message queues or batch ETL processes.

Ontology-to-scheduling feedback. Foundry ontology state informs Symphony scheduling decisions through ELIM metrics derived from ontology queries. A consumer policy specifying "route inference requests to model instances trained on the most recent ontology-validated data" queries Foundry for the latest model provenance metadata and uses the result as a scheduling constraint. The feedback loop closes the gap between operational intelligence (ontology) and compute orchestration (scheduling).

Agent-orchestrated workflows. AIP agents operating within Foundry's governed ontology invoke Symphony services as action implementations. An agent detecting a risk threshold crossing in the ontology triggers a Symphony SOAM workflow performing quantum-enhanced analysis, neuromorphic anomaly detection, and LLM narrative generation. The workflow results flow back to Foundry as ontology object updates. The agent operates within Foundry's access controls and audit framework; the compute operates within Symphony's scheduling framework. Neither layer transgresses the other's governance boundary.

C. ELIM Metrics for the Extended Stack

ELIM scripts report per-resource-type metrics enabling Symphony to make scheduling decisions informed by each resource's operational state.

All metrics feed the same Symphony scheduling algorithm through the same ELIM interface. A scheduling decision routing an inference request to a GPU with available KV cache capacity uses the same framework as a decision routing a classification task to a neuromorphic chip with low power utilization. The ontology receives objects from all compute tiers without distinguishing the scheduling mechanisms governing each tier's workload placement.

D. GPFS as Cross-Paradigm Coordination Substrate

GPFS connects all compute tiers and the Foundry ontology through a unified storage fabric [20]. NVIDIA GPUDirect Storage enables direct data transfer between GPU memory

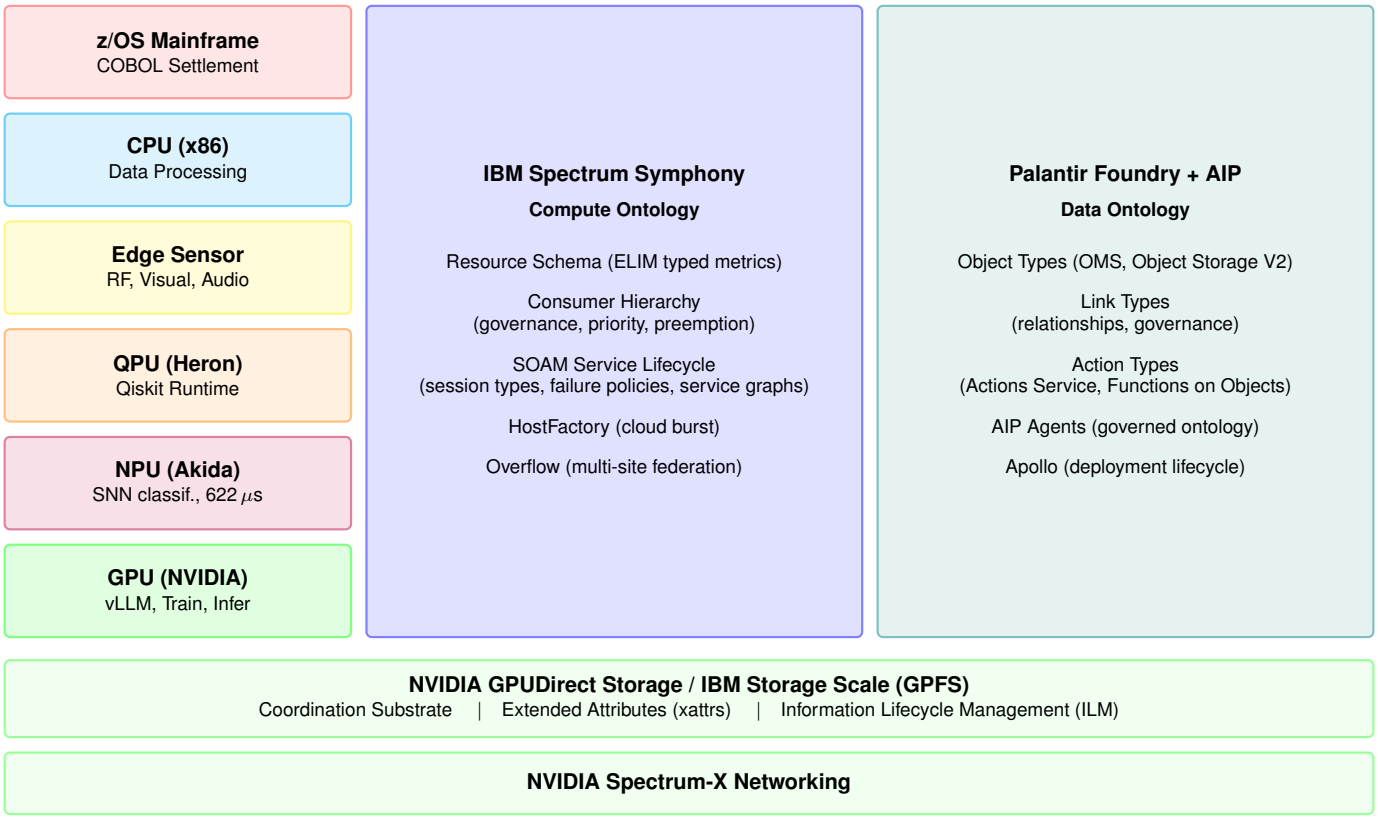

Fig. 1. Extended AI OS Architecture. Compute tiers feed Symphony's compute ontology through typed ELIM metrics. Symphony and Foundry's data ontology operate as parallel ontological layers. GPFS provides the coordination substrate across all tiers and both ontologies.

and GPFS without CPU intermediation, eliminating the CPU bottleneck that would otherwise constrain GPU-intensive inference and training workloads operating against shared storage [21]. The role of GPFS in the extended architecture mirrors its role in the quantum-centric heterogeneous orchestration architecture [2]: GPFS carries heterogeneous artifacts between tiers without requiring a common data model, a message broker or an intermediate serialization layer.

The artifacts traversing GPFS in the extended architecture span the full range of computational paradigms:

- **Neuromorphic observations:** 128-byte classified observations from AKD1000 spiking neural networks, carrying classification ID, confidence, threat score, and sensor modality
- **LLM outputs:** Reflection narratives, semantic analyses, and wisdom crystals produced by Nemotron-3-Super-120B
- **GPU tensors:** Model weights, KV cache state, and training checkpoints
- **Quantum artifacts:** Circuit definitions, bitstring samples, and quantum-random key material
- **Foundry exports:** Ontology snapshots, object type definitions, and dataset schemas exported from Foundry for local processing
- **Sensor streams:** RF captures, acoustic spectrograms, visual frames, and BLE telemetry from edge sensors

Extended attributes (xattrs) on GPFS files carry per-artifact metadata enabling downstream services to discover, filter, and process artifacts without knowledge of the producing tier's internal formats [20]. The metadata is queryable through standard filesystem interfaces and provides the provenance tracking Foundry's audit requirements demand.

E. Multi-Cloud and Hybrid Deployment

The extended architecture inherits multi-cloud capability from both Palantir and Symphony. Palantir's Apollo deploys Foundry and AIP across AWS, Azure, GCP, private cloud, and air-gapped environments through pull-based release channels [5]. Symphony's HostFactory provisions compute resources across IBM Cloud, AWS, and on-premise infrastructure within the same scheduling domain [3]. The combination enables deployment topologies the AIOS-RA alone cannot express:

Sovereign on-premise. All components run on-premise: Symphony, GPFS, GPU compute, neuromorphic edge, Foundry (via Apollo). Data never leaves the customer's infrastructure perimeter.

Hybrid cloud-edge. Foundry runs on AWS (Palantir-hosted). Symphony manages on-premise GPU and neuromorphic compute. HostFactory bursts inference workloads to IBM

TABLE I
SAMPLE ELIM METRICS BY RESOURCE TYPE IN THE EXTENDED ARCHITECTURE.

Resource Type	ELIM Metrics	Ontology Relevance
GPU (NVIDIA)	`gpu_util`, `gpu_mem`, `kv_cache_util`, `model_tier`, `tokens_per_sec`, `inference_latency_p95`	Model inference feeding ontology objects
NPU (AKD1000)	`npu_model_loaded`, `npu_inference_latency`, `npu_power_mw`, `npu_spike_rate`, `npu_confidence`	Edge perception feeding observation objects
QPU (Heron R3)	`qpu_fidelity`, `qpu_queue_depth`, `qpu_calibration_age`	Quantum-enhanced features, cryptographic keys (demonstrated in [2])
Edge Sensor	`sensor_health`, `signal_strength`, `observation_rate`, `modality_type`	Raw sensor data feeding edge classification
CPU (x86)	`cpu_util`, `mem_avail`, `data_pipeline_throughput`	Data processing, transformation, enrichment
Mainframe	`zos_initiators`, `zos_batch_queue`	Settlement, regulatory reporting (demonstrated in [2])

Cloud A100 instances during peak demand. Edge sensors feed the on-premise Symphony cluster through GPFS. The ontology in Foundry receives objects from both on-premise and cloud compute tiers.

Federated multi-site. Symphony Overflow federates scheduling across multiple sites, each running its own GPU and neuromorphic compute. Foundry provides the unified ontology layer across all sites. A classification event at Site A triggers an LLM analysis at Site B (where the larger GPU is available) and writes the result to the shared Foundry ontology accessible from both sites.

The deployment flexibility addresses the practical constraint that many organizations, particularly in financial services and defense, cannot commit to a single cloud provider or a single deployment model. The architecture accommodates hybrid deployments where some components run on AWS (Foundry), some run on IBM Cloud (GPU burst capacity), and some run on-premise (neuromorphic edge, GPFS, sensitive data processing).

V. DEMONSTRATIONS

A. Overview

Four working demonstrations validate the extended architecture as existence proofs. Each demonstration exercises a different aspect of the Symphony-Foundry-NVIDIA integration and provides evidence that heterogeneous compute orchestration extends the AIOS-RA's capabilities beyond what a GPU-only Kubernetes substrate provides. The demonstrations prove the architecture is implementable. The production evidence from the components, Symphony managing billions of tasks daily in financial services, GPFS scaling to exabytes, Foundry supporting tens of billions of ontology objects, proves the architecture scales. The specific hardware on which the demonstrations run is immaterial to the architectural argument because the compute ontology operates identically regardless of the GPU model, network bandwidth or node count it governs.

B. SymWisdom: Autonomous Ontology Construction

SymWisdom demonstrates a capability no GPU-only architecture can replicate: autonomous ontology construction through multi-domain neuromorphic perception and LLM reflection [6]. Ten BrainChip AKD1000 spiking neural network processors perceive reality across seven sensor modalities in continuous time, each node running a trained spiking neural network model matched to its sensor type. GPFS stores accumulated perceptual state as a shared memory-mapped substrate, with state files enabling all nodes to read and write simultaneously. Nemotron-3-Super-120B reflects on the accumulated perceptual experience every 45 seconds, identifying patterns, anomalies, and emergent behaviors. Crystallized insights, concepts the system discovers through its own perceptual experience rather than through human instruction, are written to Palantir Foundry as ontology objects.

The sensor diversity is architecturally significant but representative rather than prescriptive. The specific modalities and sensor hardware listed below serve as a demonstration fleet. Any sensor capable of producing a digitized signal can substitute for any entry in the table, and the trained spiking neural network model on each AKD1000 can be replaced with a model matched to the replacement sensor's modality. The combination provides multi-domain environmental awareness across the electromagnetic, acoustic, and proximity domains.

Seven distinct modalities span L-band satellite RF (1.6 GHz), UHF public safety (453 MHz), VHF weather (162.5 MHz), visible light (400–700 nm wavelength), acoustic (20 Hz – 20 kHz), BLE proximity (2.4 GHz), and WiFi/BLE proximity detection across all ten active nodes. The coverage extends from audio frequencies through visible light through L-band RF on commodity hardware consuming under 200 watts total. Each 128-byte classified observation produced by an AKD1000 carries a sensor modality tag and the emergence engine fuses observations across modalities to detect patterns no single modality reveals independently.

Symphony's resource management scales the sensor fleet horizontally without architectural modification. The ten-node demonstration fleet is a convenience of available hardware, not an architectural constraint. Symphony's SOAM framework allocates Service Instances to hosts dynamically, so expanding from ten nodes to one thousand requires only adding

TABLE II
SENSOR FLEET: PER-NODE MODALITY ASSIGNMENTS.

Node	Modality	Sensor Hardware	Spectrum/Domain	AKD1000 Model
Node 1	Iridium satellite RF	bladeRF xA4, L-band patch antenna	1616–1626.5 MHz (L-band)	RF burst classifier
Node 2	Visual	Reolink 4MP RTSP camera	Visible light	Vehicle/person classifier
Node 3	Visual	Reolink 4MP RTSP camera	Visible light	Vehicle/person classifier
Node 4	Visual	Reolink 4MP RTSP camera	Visible light	Vehicle/person classifier
Node 5	Emergency dispatch RF	RTL-SDR, 453.6 MHz	UHF public safety band	Dispatch event classifier
Node 6	Iridium satellite RF	bladeRF xA4, L-band patch antenna	1616–1626.5 MHz (L-band)	RF burst classifier
Node 7	NOAA weather radio	RTL-SDR, 162.5 MHz	VHF weather band	Weather alert classifier
Node 8	BLE/IoT + WiFi proximity	ESP32 Heltec V4 (LoRa relay)	2.4 GHz BLE, WiFi	Proximity classifier
Node 9	Acoustic	Camera audio, mel spectrogram	20 Hz – 20 kHz audio	Acoustic event classifier
Node 10	Visual	Reolink 4MP RTSP camera	Visible light	Vehicle/person classifier

hardware and registering the new hosts in the appropriate resource group. EGO discovers each new node's ELIM metrics automatically and the scheduling framework distributes sensor workloads across the expanded fleet. The architecture provides not only a hospitable ontology for heterogeneous compute but one that scales linearly with the operational footprint.

The multi-domain perception capability differentiates the extended architecture from any GPU-only alternative. A GPU-based perception system can process video and audio at server-class power and latency. Neuromorphic perception extends the coverage to RF spectrum analysis, satellite burst detection, and proximity sensing at milliwatt power per node, enabling deployment at locations where server-class hardware is impractical. The ontology built from multi-domain perception captures operational reality with a richness no single-modality system achieves.

The architecture comprises six layers:

TABLE III
SYMWISDOM ARCHITECTURE LAYERS.

Layer	Technology	Function
Perception	10x AKD1000 on Nodes 1–10	Multi-domain sensory experience, 7+ modalities
Shared State	GPFS mmap'd state	All nodes read/write simultaneously
Experience	GPFS /data/experience/	Lived memory with temporal structure
Reflection	Nemotron-3-Super-120B on vulcan-hs	Deep thought every 45 seconds
Wisdom	Palantir Foundry	Crystallized understanding as ontology objects
Voice	NemoClaw agent on vulcan-hs	The system's interface to operators

In a 40-hour operational period, SymWisdom produced 1,646 reflections and 6 wisdom objects with zero human guidance. The system named its own patterns, creating ontology object types based on discovered regularities in the perceptual data rather than on human-designed schemas. A follow-up extended the system with homomorphic encryption (OpenFHE) and IBM Quantum for cryptographic key generation across 15 models, producing 3,845 reflections and the first cross-domain wisdom object bridging patterns observed across multiple sensor modalities.

The demonstration establishes that the AIOS-RA's ontology layer, designed to receive human-structured data through ETL pipelines and human-designed schemas, can receive autonomously discovered knowledge from a heterogeneous compute substrate in which neuromorphic perception and LLM reflection collaborate through GPFS-mediated experience sharing. The capability extends Foundry's Ontology Intelligence from a tool that discovers structure in existing data to a system that discovers structure in reality through continuous perception.

C. Edge-to-Ontology: Multi-Modal Sensor Fusion System

The SymPalantir-Akida sensor fusion system demonstrates the full edge-to-ontology pipeline under Symphony orchestration [7]. The same ten-node sensor fleet described in Section V-B, spanning L-band satellite RF, visible light, UHF dispatch, VHF weather, BLE/IoT proximity, acoustic, and WiFi proximity modalities, produces thousands of classified observations per hour. Each node's AKD1000 classifies raw sensor input at 622 microseconds and 30 milliwatts, discarding non-events before any data traverses the network. Symphony's emergence engine fuses surviving observations across modalities, collapsing redundant detections through spatial and temporal correlation. A vehicle detection on the visual modality (Node 2) correlating with an acoustic engine signature (Node 9) and a BLE transponder detection (Node 8) within a 60-second temporal window and shared spatial proximity collapses three independent observations into one confirmed event. The emergence engine requires convergence from a minimum of three independent modalities within the temporal window before confirming an event, ensuring confirmed events withstand independent witness rather than relying on any single sensor's classification. Granite LLM vali-

dates semantic consistency, rejecting physically implausible or referentially inconsistent observations. Validated intelligence records are written to Palantir Foundry as typed ontology objects (VehicleEvent, AcousticEvent, RfEvent, EldDetection, IridiumFleetTracker, EmergencyDispatchEvent) linked to Target fusion objects through Foundry's link type system.

The cross-modal fusion is the key architectural contribution. Palantir's SIP can deploy models to individual sensors and collect telemetry from each but the fusion logic, the emergence engine determining when independent sensor observations converge on a single real-world event, operates within Symphony's SOAM service graph rather than within a CI/CD pipeline. The scheduling framework manages the fusion in real time, consuming ELIM metrics from each node (classification confidence, spike rate, observation rate) and routing observations to the emergence engine based on resource availability and temporal urgency.

The system achieves a 155:1 data distillation ratio: 287,000 edge events become 1,847 Foundry records per hour. The distillation occurs at the compute edge and within the orchestration layer rather than at the ontology ingest point. Symphony SOAM manages the full pipeline from neuromorphic classification through cross-modal fusion through semantic validation through ontology ingestion as a single service graph.

TABLE IV
EDGE-TO-ONTOLOGY PIPELINE PERFORMANCE METRICS.

Stage	Latency	Technology
AKD1000 inference	622 μs	Spiking neural network
Emergence engine fusion	milliseconds	Symphony SOAM on x86
Granite semantic validation	1.2 seconds	vLLM on NVIDIA GPU
Foundry ontology ingestion	seconds	REST API
End-to-end (edge to Foundry)	7 seconds (demo), $<$ 4 min (target)	Full pipeline

The demonstration addresses a problem the AIOS-RA's GPU-only compute model cannot solve. Sensor data flooding, exemplified by the 25th Infantry Division's experience of "thousands of data objects with no way to control the flow from Palantir" during Lightning Surge 1, arises when raw sensor data enters the ontology without edge-side filtering. GPU-based filtering requires server-class hardware at the sensor location, consuming kilowatts per classification. Neuromorphic filtering at 30 milliwatts per chip enables classification at the sensor itself and the 155:1 distillation ratio ensures the ontology receives intelligence rather than noise.

D. SymPalantir RAG: Computational Diversity as Security

The SymPalantir RAG demonstration provides initial evidence through a targeted test scenario that computational diversity across fundamentally different hardware architectures can produce security properties beyond what single-paradigm architectures achieve [8]. The system protects a RAG knowledge base against document poisoning through four trust layers, each operating on a different computational substrate:

1) **Cryptographic signature verification** (CPU): Sub-millisecond verification using quantum-random keys from IBM Quantum
2) **Multi-party approval** (Symphony orchestration): Workflow enforcement requiring multiple authorized signatories
3) **Semantic review** (GPU, Nemotron-3-Super-120B): Linguistic contradiction detection at 40–45 tokens per second depending on prompt complexity
4) **Neuromorphic anomaly detection** (AKD1000): Geometric pattern classification at 9 milliseconds with 100% accuracy on poisoned documents

Against a poisoning scenario of three fabricated documents claiming false financial figures, the SymPalantir RAG stack achieved 0% attack success across all test runs, compared to 15% residual semantic injection success for a five-layer GPU-only defense stack. The test scenario is narrow, comprising three fabricated financial documents against a 19-document knowledge base. The 0% result demonstrates the architectural principle of computational diversity but should not be generalized to all poisoning scenarios without broader adversarial testing across document types, knowledge base sizes, and attack sophistication levels. The difference arises because the neuromorphic anomaly detector operates on fundamentally different computational principles than the LLM-based semantic review. An adversarial document crafted to fool a transformer gains no advantage against a spiking neural network classifier operating on different silicon with a different computational model.

Palantir Foundry serves as the immutable audit trail. Every submission, every layer verdict, and every rejection reason is written to Foundry's ontology across six object types and five link types with AES-256-GCM encrypted audit records. The audit trail is externally hosted, immutable and queryable through Foundry's access controls. An attacker compromising the entire compute stack cannot alter the record of what was rejected.

The demonstration extends the AIOS-RA's security model from Rubix's zero-trust network security to computational diversity at the workload level. Rubix protects the infrastructure perimeter. Heterogeneous compute protects the intelligence pipeline. The two security models are complementary and orthogonal.

E. SymCognitive: AI-Enabled Financial Ontology

SymCognitive demonstrates AI-enabled ontology creation for financial services, extending Foundry's ontology with entities and relationships discovered through LLM analysis of financial data [9]. The system processed stock price data, corporate filings, and market indicators for 37 financial firms, discovering 1,804 entities and 2,690 relationships. Entity types include Company, Sector, MarketRegime, CorrelationCluster, and RiskFactor. Relationship types include sector membership, correlation links, regime transitions, and risk exposures.

The ontology construction pipeline runs under Symphony SOAM management:

1) **Data ingestion** (Symphony service): Load financial data from GPFS, normalize formats, compute derived features
2) **Entity extraction** (Symphony service, GPU): Nemotron-3-Super-120B identifies entities from unstructured financial text
3) **Relationship discovery** (Symphony service, GPU): LLM-based analysis identifies relationships between discovered entities
4) **Ontology construction** (Symphony service): Create Foundry object types, link types, and action types based on discovered entities and relationships
5) **Validation** (Symphony service, NPU): Neuromorphic anomaly detection validates discovered entities against known financial patterns

The demonstration validates that Foundry's ontology, designed to be human-constructed, can be extended through AI-driven discovery under heterogeneous compute orchestration. The financial services use case directly addresses the AIOS-RA's target market, demonstrating that the extended architecture produces operational intelligence relevant to the institutions Palantir and NVIDIA serve.

VI. Evaluation

A. Architectural Comparison

The extended architecture addresses specific limitations of the AIOS-RA's Kubernetes-based compute substrate.

B. Inference Orchestration: Symphony versus Run:ai

The AIOS-RA positions Run:ai as the GPU scheduling layer within its Kubernetes substrate [1]. The comparison between Run:ai's container-level GPU scheduling and Symphony's workload-level inference orchestration illuminates the architectural gap the present paper addresses.

Symphony's ELIM reports metrics per vLLM [22] inference instance [23] through a native binary protocol operating at sub-millisecond query latency, compared to Prometheus scrape intervals of 15–30 seconds in typical Kubernetes deployments at 0.89% CPU overhead per host. ELIM imposes no architectural limit on the number of metrics an external load information script can report. The demonstration deployment reports 38 vLLM-specific metrics spanning seven categories: performance (TTFT, TPOT, end-to-end latency and their P95 variants, tokens per second, active sessions), request tracking (completed requests, prompt tokens, generation tokens, average prompt length), cache and memory (KV cache utilization, prefix cache hit rate, preemption count), queue and status (queue depth, model loaded, model name), and GPU (utilization, memory used, memory free). Additional metrics for any operational dimension can be added by extending the ELIM script without modifying Symphony itself. Run:ai provides GPU-count-level container scheduling within Kubernetes; Symphony provides real-time workload management informed by arbitrarily rich operational metrics per instance.

The distributed inference deployment implements four scoring algorithms as native Symphony SOAM services, providing an alternative to Red Hat's `llm-d` framework [24] rather than depending on it [25]: LoadAwareScorer (weight 0.40, using `vllm_queue_depth` with GPU utilization penalty), SessionAffinityScorer (weight 0.25, header-based sticky routing with LRU cache), NoHitLRUScorer (weight 0.20, distributing cold requests evenly) and ActiveRequestScorer (weight 0.15, using `vllm_active_sess`). The combined routing decision completes in approximately 10 milliseconds (5 ms for cached ELIM state query, 5 ms for scorer computation). Run:ai dispatches containers to GPU nodes; Symphony manages inference workloads within running containers based on real-time operational state, a fundamentally different level of scheduling granularity.

The semantic routing system further extends inference orchestration with capability-based routing. A KNN classifier trained on 415 samples achieves 91.1% accuracy in classifying queries across code, math, and general domains. Symphony's native arithmetic resource requirement expressions (`select(llm_score_code >= 70) order(llm_score_code)`) route queries to model instances reporting matching capability scores through ELIM. The semantic router achieves 74–81% cost savings by routing simple queries to smaller models [26], [27], [28], a capability requiring per-model operational metrics no Kubernetes-based scheduler natively provides. The system includes a mobile application built with React Native and IBM Carbon Design for real-time inference analytics.

Run:ai's architectural gaps extend beyond metric and routing differences. Run:ai supports only NVIDIA GPUs, excluding the neuromorphic, quantum, edge sensor, and mainframe resources the extended architecture integrates. The open-sourced KAI Scheduler lacks runtime GPU enforcement, meaning containers can exceed their allocated GPU limits without detection or correction (GitHub Issue #423) and GPU over-allocation triggers infinite retry loops (Issue #848) [12]. The NVIDIA-only constraint and the absence of runtime enforcement are not configuration gaps but architectural boundaries inherent in building a GPU scheduler on a container placement engine.

These capabilities are directly relevant to the AIOS-RA because AIP agents operating within Foundry's ontology invoke LLM inference as a core operation. The AIOS-RA's rate-limiting approach, metering LLM usage by tokens per minute and requests per minute per model per provider, addresses capacity management but not workload-aware routing. Symphony provides the routing intelligence the ontology-driven inference pattern demands: route a complex reasoning query from an AIP agent to the 120B model instance with available KV cache capacity and acceptable P95 latency, while routing a simple entity extraction to the 2B model at 74% lower cost.

C. Power Efficiency

The neuromorphic tier provides measurable power efficiency advantages for classification workloads feeding the

TABLE V
ARCHITECTURAL COMPARISON: AIOS-RA VERSUS EXTENDED ARCHITECTURE.

Capability	AIOS-RA (Kubernetes + SIP)	Extended (Symphony + K8s + SIP)
GPU inference scheduling	Run:ai (container-level)	ELIM metrics per instance, SOAM scorer services
Inference metric latency	15–30 s (Prometheus scrape interval)	$<$ 1 ms (ELIM native binary)
Inference cost optimization	Rate limiting (TPM/RPM)	74–81% savings via semantic routing
Neuromorphic compute	Not supported	ELIM NPU metrics, 622 μs classification
Quantum compute	Not supported	ELIM QPU metrics, HostFactory
Edge AI deployment	SIP + Apollo (CI/CD, Micro Models)	Symphony soamdeploy + ELIM real-time orchestration
Edge sensor fusion	SIP ensemble/chaining (per device)	Cross-device SOAM fusion, GPFS shared state
Sensor modality coverage	Video, radar, acoustic (SIP)	7+ modalities, multi-domain
Edge-cloud state sync	Object Peering (eventually consistent)	GPFS mmap'd state (sub-ms, simultaneous)
Cross-paradigm workflows	Application-level HTTP	SOAM service graph composition
Temporal heterogeneity	Assumed uniform	Five orders of magnitude
Autonomous ontology growth	Knowledge Nodes (content enrichment)	SymWisdom (schema generation from perception)
Business process orchestration	Orchestrator (durable, human-in-loop)	Not addressed (complementary)
Computational security diversity	Rubix perimeter + OSPs	Rubix + multi-paradigm trust verification
Resource metering	FIFO queues (vCPU/vGPU)	Consumer hierarchies, priority, preemption
Data distillation	Software filtering (SIP)	155:1 neuromorphic silicon-level distillation
Production scheduling scale	K8s cluster limits	Billions of tasks/day in FS

ontology.

The AKD1000 chip consumes 30 mW during inference but the host system (Intel N100 mini-PC) consumes approximately 15 W at the wall [18], [19]. The chip-level power advantage (67:1 to 1,667:1) represents the neuromorphic silicon's intrinsic efficiency. The system-level comparison against edge inference hardware (Jetson Orin Nano at $\sim$10 W, Coral TPU at $\sim$2 W) shows the N100 host dominates total power consumption. The architectural advantage of neuromorphic silicon at the system level is latency (622 microseconds versus milliseconds for GPU/TPU classifiers) and computational diversity (event-driven spiking inference versus synchronous tensor computation with sparsity-driven noise rejection at the silicon level) rather than raw system power. Future AKD1000 deployments on lower-power hosts or as PCIe accelerators in existing infrastructure would shift the system-level comparison toward the chip-level ratios.

The power differential is significant for edge deployments where the AIOS-RA's Blackwell Ultra GPUs (consuming hundreds of watts each) cannot be deployed. Neuromorphic classification at milliwatt power enables perception at the sensor edge, feeding the Foundry ontology with classified observations from locations where GPU-based classification is impractical due to power, cooling, or physical space constraints.

D. Data Distillation Efficiency

The 155:1 distillation ratio achieved by the SymPalantir-Akida system addresses a quantitative problem the AIOS-RA does not solve. An ontology receiving 287,000 raw sensor events per hour faces storage, indexing, and query performance challenges. An ontology receiving 1,847 distilled intelligence records per hour operates within manageable bounds while preserving the operational intelligence the raw data contains.

The distillation occurs through three mechanisms managed by Symphony's SOAM service graph:

1) **Neuromorphic filtering** (AKD1000): Spiking neural networks classify raw sensor input, discarding non-events at the edge
2) **Emergence engine fusion** (Symphony SOAM): Temporal and spatial correlation collapses redundant observations across sensor modalities
3) **Semantic validation** (Granite LLM): Linguistic analysis rejects physically implausible or referentially inconsistent observations

Each mechanism operates on a different computational substrate and the combination produces distillation no single mechanism achieves alone.

E. Security Properties

The computational diversity argument established by SymPalantir RAG extends the AIOS-RA's security model in a dimension Rubix does not address. Rubix provides infrastructure-level security: identity verification, encrypted communication, access control, and audit logging [1], [5]. Computational diversity provides workload-level data integrity verification through fundamentally different computational paradigms, a security dimension orthogonal to infrastructure perimeter controls.

The distinction matters because the AIOS-RA's ontology is the operational decision layer. A compromised ontology produces compromised decisions regardless of how secure the infrastructure perimeter is. Rubix prevents unauthorized access to the ontology. Computational diversity prevents poisoned data from entering the ontology through authorized channels. The two security properties are orthogonal and complementary.

TABLE VI
POWER EFFICIENCY COMPARISON ACROSS CLASSIFICATION HARDWARE.

Task	GPU (server-class)	NPU (AKD1000 chip)	NPU (N100 system)	Ratio (chip)	Ratio (system)
Single classification	∼50 W	30 mW	∼15 W	1,667:1	3.3:1
Classification latency	∼10 ms	622 μs	622 μs	16:1	16:1
Edge classif. (Jetson Orin Nano)	∼10 W	30 mW	∼15 W	333:1	0.67:1
Edge classif. (Google Coral TPU)	∼2 W	30 mW	∼15 W	67:1	0.13:1

F. Scalability

The extended architecture inherits scaling properties from its components. Symphony manages billions of tasks daily at the largest financial institutions in the world [3], [4]. Foundry's Object Storage V2 supports tens of billions of objects per type [10]. GPFS scales to exabytes across thousands of nodes [20]. The demonstrations validate architectural integration; the scaling evidence comes from each component's independent production history. The compute ontology (resource schema, consumer hierarchy, SOAM lifecycle) is defined independently of the number of resources it governs and scales without architectural modification.

The neuromorphic tier scales horizontally by adding AKD1000 nodes to the Symphony scheduling domain. Adding 100 or 1,000 neuromorphic nodes requires only ELIM registration and SOAM service deployment, not architectural modification.

VII. DISCUSSION

A. Ontology-Native Intelligence

The demonstrations establish a capability the AIOS-RA enables but does not fully exploit: ontology-native intelligence in which the ontology grows through computational discovery rather than solely through human design. Foundry's ontology is designed to receive structured data through ETL pipelines and human-designed schemas. The SymWisdom and SymCognitive demonstrations show the ontology receiving knowledge from autonomous computational processes operating across heterogeneous compute substrates.

The capability has implications beyond the demonstrations. An ontology growing through neuromorphic perception discovers patterns human observers miss because the temporal resolution of neuromorphic classification (622 microseconds) exceeds human perceptual bandwidth by orders of magnitude. An ontology growing through multi-domain sensor coverage, spanning L-band satellite RF, UHF public safety, VHF weather, visible light, acoustic, BLE proximity, and WiFi proximity detection, captures correlations invisible to any single-modality system. An Iridium satellite burst correlating with a vehicle detection correlating with an acoustic signature is a pattern only a multi-modal perception system can discover and the correlation emerges at the scheduling layer through Symphony's SOAM service graph rather than through post-hoc data analysis. An ontology growing through LLM reflection discovers relationships across documents and datasets human analysts cannot process at comparable scale. An ontology growing through quantum-enhanced feature encoding captures correlations classical feature engineering cannot represent. Each compute paradigm and each sensor modality contributes perceptual capabilities the others lack and the ontology integrates contributions from all paradigms and modalities without requiring a common perceptual model.

The field-deployed engineer model, in which value is demonstrated to clients in five days or less rather than through multi-month proof-of-concept engagements, aligns with the rapid ontology construction the demonstrations enable. The SymPalantir-Akida sensor fusion system was built in five days. The SymCognitive financial ontology discovery processed 37 firms' data in a single operational run. The SymWisdom system produced its first wisdom objects in 40 hours. The architecture enables rapid value demonstration because the ontology grows from data rather than waiting for schema design.

B. Complementary Architecture, Not Competitive

The extended architecture proposed in the present paper is complementary to the AIOS-RA rather than competitive with it. The distinction is important and deserves explicit articulation.

Palantir Foundry is the best enterprise AI platform for application-layer reasoning and data governance. Foundry's multi-user, multi-role security model, its ontology-mediated reasoning framework, its Orchestrator for durable business workflows, its 150+ data connectors, and its Knowledge Nodes for autonomous knowledge capture are capabilities no HPC scheduling framework provides or should attempt to provide. Apollo's edge deployment pipeline with DISA IL5/IL6 authorization is world-class for classified environments. SIP's sensor fusion with Micro Model chaining, hot-swap and ensemble outputs is a production-grade edge inference platform.

Palantir's own engineering philosophy articulates the ontology's role as a "defragmentation layer" for enterprise architecture, consolidating per-application glue code into a shared semantic layer where AI systems interact through the same interfaces as human users. The philosophy explicitly requires human construction of the ontology schema: domain experts define object types, link types, and action types; AI operates within the schema humans provide. Symphony extends Foundry into capabilities the OOSD philosophy does not contemplate: autonomous ontology schema generation

from neuromorphic perception (SymWisdom), real-time workload scheduling across heterogeneous hardware based on per-resource operational metrics, closed-loop direction of physical sensors through SOAM service graphs and GPFS-mediated cross-node state sharing at sensor rate.

The complementarity extends to security. Palantir's hallucination mitigation operates in three layers: Ontology Augmented Generation grounds LLM responses in validated data, logic delegation routes deterministic computation to typed Ontology Functions rather than LLM approximation, and human-AI teaming stages AI-suggested actions for expert review. The three layers form a trust gradient addressing knowledge gaps, reasoning errors, and judgment errors. The SymPalantir RAG demonstration addresses a fourth failure mode the trust gradient does not cover: data poisoning of the grounding data itself. Palantir's first layer (OAG) assumes the Ontology data is trustworthy. The four-layer trust chain (quantum-random signature verification, multi-party approval, semantic firewall, neuromorphic anomaly detection) protects the data before any LLM processes it. The two security models address different threat vectors and combine without overlap.

The integration points are clean. Palantir handles business logic, multi-user reasoning, and enterprise governance. Symphony handles edge perception, hardware orchestration, and real-time compute dispatch. Results from Symphony project into Foundry's ontology through the observation-to-ontology pipeline. Foundry agents direct Symphony's cluster through MCP-mediated service invocations. Neither layer transgresses the other's governance boundary.

C. The Palantir Field-Deployed Engineer Model

Palantir's commercial success stems partly from its Field-Deployed Engineer model, in which Palantir engineers embed with customers to build solutions directly on Foundry and AIP. The extended architecture enhances the model by providing heterogeneous compute capabilities the field-deployed engineer can leverage without requiring additional hardware procurement for each engagement.

A field-deployed engineer arriving at a financial institution with an existing Symphony grid can deploy neuromorphic classification, LLM inference and ontology construction services as SOAM services on the existing infrastructure. The neuromorphic hardware cost is negligible relative to enterprise compute budgets. The Symphony scheduling framework is already managing the institution's compute workloads. Adding Foundry as the ontology layer and AKD1000 boards as the neuromorphic perception layer extends the existing infrastructure rather than requiring a parallel deployment. Neuromorphic chips can also offload classification tasks from existing enterprise inference pipelines, reducing GPU cost per classification while freeing GPU capacity for workloads requiring transformer-scale computation. Alternatively, additional neuromorphic nodes extend the institution's capability into domains conventional inference hardware does not serve efficiently, including real-time sensor classification, anomaly detection at the edge, and continuous environmental monitoring.

The NVIDIA AI Factory architecture presents a further opportunity the original AIOS-RA likely did not consider. DGX and HGX systems contain substantial CPU capacity alongside their GPU resources, and Symphony can schedule SOAM-based services on those CPUs as readily as on any other compute resource. Data preprocessing, ontology construction, semantic validation, emergence logic, and other non-GPU workloads can run as SOAM services on the CPU cores of machines the AIOS-RA treats exclusively as GPU infrastructure. Symphony's resource-agnostic scheduling enables Foundry's ontology pipeline to harness the full computational capacity of an AI Factory rather than leaving CPU resources idle while GPUs process inference requests.

D. IBM Cloud as Additional Hosting Platform

The extended architecture's multi-cloud capability creates an opportunity for IBM Cloud as a hosting platform for Palantir Foundry. Palantir currently deploys on AWS, Azure, GCP and on-premise through Apollo [5]. IBM Cloud provides NVIDIA GPU instances, GPFS as a managed service and Symphony as a managed scheduling platform. The combination would enable Palantir deployments on IBM Cloud with native access to GPFS coordination and Symphony orchestration, capabilities no other cloud provider offers natively.

The proposition to Palantir is operational: hosting Foundry on IBM Cloud provides integrated access to the orchestration and storage layers the extended architecture demonstrates, without requiring the customer to deploy and manage Symphony and GPFS independently. The proposition to IBM is strategic: Palantir's customer base represents the enterprise segment most likely to adopt heterogeneous compute orchestration for operational intelligence applications.

E. Relationship to the AI Factory Architecture

The extended AIOS-RA and the dual-scheduler AI factory architecture presented in a companion paper [29] are complementary reference architectures addressing different market segments. The AI factory paper addresses large-scale training and inference on NVIDIA GPU infrastructure with LSF and Symphony as the scheduling layer. The present paper addresses ontology-driven operational intelligence with Foundry and Symphony as the intelligence and orchestration layers. The two architectures share Symphony as the inference orchestrator and GPFS as the coordination substrate but serve different primary use cases.

An integrated deployment combining both architectures would use LSF for model training, Symphony for inference and heterogeneous compute orchestration, GPFS as the unified storage fabric, and Foundry as the ontology layer receiving intelligence from all compute tiers. The integrated architecture represents the complete stack: training (LSF), inference and heterogeneous orchestration (Symphony), storage and coordination (GPFS), and operational intelligence (Foundry), grounded in NVIDIA GPU compute with neuromorphic, quantum, and mainframe augmentation.

F. Limitations

Several limitations qualify the claims of the present work. The Palantir Foundry integration uses REST APIs and SDK calls rather than a native Symphony-Foundry connector. A production deployment would benefit from tighter integration, potentially through the Ontology MCP protocol, enabling Foundry operations to participate as first-class SOAM service invocations with transactional guarantees.

The neuromorphic demonstrations use first-generation AKD1000 silicon. BrainChip's AKD1500, currently in manufacturing with expected availability in Q3 2026, will improve throughput and expand the range of neuromorphic workloads available for ontology-feeding perception services. The demonstrations validate the architectural integration; the production capability will improve with next-generation hardware.

The SymWisdom autonomous ontology construction is demonstrated in a research context. Production deployment of autonomous ontology growth requires governance mechanisms ensuring the system does not create ontology objects violating data quality standards, compliance requirements or organizational policies. Foundry's existing governance framework (role-based, marking-based, and purpose-based access controls) provides the foundation but extending governance to autonomous computational agents creating ontology objects at machine cadence is an open challenge.

GPFS as the coordination substrate requires network connectivity between nodes. The architecture addresses connectivity loss through the same pattern Palantir's Embedded Ontology employs for disconnected operation. The neuromorphic inference path operates entirely in `/dev/shm` shared memory with zero GPFS dependency. An asynchronous flush mechanism buffers state locally and writes to GPFS when available, with hybrid logical clocks maintaining causal ordering for reconnection replay. The system treats GPFS disconnection as an operational mode rather than an error condition. Palantir's Embedded Ontology handles the same scenario identically: local CRUD operations continue with millisecond latency during disconnection and Object Peering reconciles state when connectivity restores. The two approaches are architecturally parallel.

The security claims for computational diversity are architectural rather than formally verified. The 0% attack success rate was measured against a specific poisoning scenario with a specific set of fabricated documents. Formal security analysis of computational diversity as a defense property, including adversarial models spanning multiple computational paradigms simultaneously, remains a direction for future work.

Symphony does not hold FedRAMP authorization or DISA IL5/IL6 accreditation comparable to Palantir's Apollo and Rubix. The accreditation gap is significant for defense and federal deployments. However, Symphony's production deployments at major financial institutions operate under regulatory regimes (OCC, FINRA, SEC, PRA, FCA) whose security audit requirements for trading and risk infrastructure are arguably as stringent as federal certification programs. IBM Cloud holds FedRAMP, SOC 2 Type II, PCI DSS Level 1, ISO 27001, FISMA, and DoD DISA certifications [30], [31] and IBM Cloud for Financial Services provides automated controls aligned with financial regulatory requirements. The accreditation pathway for the extended architecture would leverage IBM Cloud's existing certifications for the orchestration and storage layers.

VIII. Future Work

Five directions for future work emerge from the present architecture.

Native Symphony-Foundry integration through the Ontology MCP protocol would enable Foundry ontology operations to participate as typed SOAM service invocations with ELIM-reported ontology health metrics (object count, ingestion rate, query latency). The integration would close the observation-to-ontology pipeline into a fully managed SOAM service graph with end-to-end scheduling visibility.

Symphony as Kubernetes meta-orchestrator would directly address the AIOS-RA's compute substrate limitation by placing EGO above the Kubernetes scheduling layer. In this architecture, Kubernetes clusters become resource groups in EGO's resource topology, with EGO making pod placement decisions based on real-time ELIM metrics rather than Kubernetes' static declared resource requests. Consumer hierarchies govern resource allocation across platform services, Foundry analytics, emergence pipelines, and user workloads, all running on the same Kubernetes fleet. Neuromorphic predictive resource management, in which AKD1000 spiking neural networks classify infrastructure telemetry at sub-millisecond latency, detects demand regime changes before they manifest as scheduling failures, enabling Symphony to act proactively rather than reacting to Kubernetes' after-the-fact autoscaler. HostFactory manages Kubernetes cluster lifecycle, including provisioning, scaling, and cross-cluster node migration. The architecture enables multiple platform tenants to run as governed consumers within one Symphony scheduling domain, each receiving guaranteed resource shares with policy-driven preemption under a single EGO resource broker that both LSF and Symphony already share.

Extension of the SymWisdom autonomous ontology construction to federated multi-site deployments would enable distributed neuromorphic perception across geographically separated sites feeding a shared Foundry ontology through Symphony Overflow. Each site contributes local perceptual experience; the LLM reflection layer discovers cross-site patterns invisible to any single site's perceptual substrate.

Integration with the `llm-d` distributed inference framework within the Symphony-Foundry pipeline, or equivalent functionality implemented natively in SOAM through distributed LLM technologies, would enable disaggregated prefill-decode inference for ontology-feeding LLM services [24], [32]. The prefill phase (processing context) and decode phase (generating output) would operate as separate SOAM services with independent ELIM metrics, enabling the scheduler to optimize each phase independently for ontology ingestion throughput.

The existing Symphony-native scorer implementation provides an alternative path to `llm-d` for organizations requiring workload management beyond container dispatch. Cross-engine KV cache sharing [32] would enable prefill and decode services to share cached context without redundant computation.

Formal security analysis of computational diversity as a defense property would establish theoretical bounds on the attack complexity required to simultaneously defeat multiple independent computational paradigms. The analysis should model adversarial capabilities across transformer-based, neuromorphic and homomorphic computational substrates and quantify the security advantage of paradigm diversity relative to depth-only defenses within a single paradigm.

IX. Conclusion

The Sovereign AI Operating System Reference Architecture correctly identifies that operational intelligence requires both structured data (Palantir Foundry's ontology) and accelerated computing (NVIDIA's GPU stack). The present paper extends the architecture with a third pillar: heterogeneous compute orchestration (IBM Spectrum Symphony) enabling compute resources beyond GPUs to participate in the operational intelligence pipeline.

Four working demonstrations validate the extension. SymWisdom produces autonomous ontology growth through multi-domain neuromorphic perception across seven sensor modalities and LLM reflection, a capability no GPU-only architecture can replicate at equivalent power efficiency or spectral coverage. The SymPalantir-Akida sensor fusion system achieves 155:1 data distillation from edge events to Foundry records through cross-modal neuromorphic fusion at 30 milliwatts per chip, with real-time emergence logic operating at the scheduling layer rather than within a CI/CD deployment pipeline. SymPalantir RAG provides initial evidence that computational diversity across fundamentally different hardware architectures can produce security properties beyond what single-paradigm architectures achieve, extending Rubix's infrastructure perimeter security with workload-level trust verification. SymCognitive discovers 1,804 entities and 2,690 relationships from financial data through AI-enabled ontology construction.

The extended architecture maintains all existing AIOS-RA components and respects the distinct roles each plays. Palantir Foundry provides the ontology. NVIDIA GPUs provide the accelerated compute. Kubernetes with Rubix provides the infrastructure substrate and zero-trust security. Apollo manages deployment lifecycle. Symphony adds the heterogeneous compute orchestration layer the existing stack does not provide, enabling neuromorphic, quantum, edge, and mainframe resources to participate as peer compute tiers alongside GPUs within the governed ontology framework. GPFS provides the coordination substrate connecting all compute tiers to each other and to Foundry.

The implementation lineage enabling the orchestration layer spans 39 years, from Zhou's 1987 load index through Platform Computing and IBM Spectrum Symphony. The same scheduling principle governing the quantum-centric heterogeneous orchestration architecture governs the Palantir-NVIDIA extension: measuring each resource according to its own nature and scheduling on the basis of those measurements within a unified domain. The ontology receives intelligence from every compute paradigm, spanning L-band satellite RF through visible light through acoustic through proximity sensing. The scheduler manages every compute paradigm. Neither layer requires the resources within its domain to conform to a common abstraction. Unity without uniformity, distinction without isolation, from the sensor edge to the ontology layer.

References

[1] Palantir Technologies and NVIDIA, "Palantir and NVIDIA team to deliver sovereign AI operating system reference architecture," https://investors.palantir.com/news-details/2026/Palantir-and-NVIDIA-Team-to-Deliver-Sovereign-AI-Operating-System-Reference-Architecture/, Mar. 2026, see also: NVIDIA Newsroom, "NVIDIA and Palantir Team Up to Operationalize AI".

[2] K. D. Johnson, "High performance quantum-centric supercomputing: A working implementation of heterogeneous orchestration across QPU, NPU, GPU, CPU, and other tiers," Technical Paper, Mar. 2026.

[3] IBM, "IBM spectrum symphony documentation," https://www.ibm.com/docs/en/spectrum-symphony, including Service-Oriented Architecture Middleware (SOAM), External Load Information Manager (ELIM), HostFactory, Overflow, and Consumer Hierarchies.

[4] D. Quintero *et al.*, *IBM Platform Computing Solutions*, ser. IBM Redbook SG24-8073. IBM, Dec. 2012.

[5] Palantir Technologies, "Apollo: Introduction," https://www.palantir.com/docs/apollo/core/introduction, see also: "Edge AI," https://www.palantir.com/offerings/edge-ai/; "Edge AI Whitepaper," 2022; "Manufacturing with the Connected Edge," Palantir Blog, Mar. 2026.

[6] K. D. Johnson, "SymWisdom: The experiencing LLM," 2026, demonstration repository. Ten BrainChip AKD1000 processors, Nemotron-3-Super-120B, IBM Spectrum Symphony, Palantir Foundry. 1,646 reflections, 6 wisdom objects in 40 hours.

[7] ——, "Edge-to-ontology: AKD1000 NPU fleet, symphony, palantir foundry," Feb. 2026, demonstration repository (sympalantir-akida). 155:1 data distillation ratio, 7 sensor modalities, 10 AKD1000 chips.

[8] ——, "SymPalantir RAG: Why RAG security requires heterogeneous compute," https://kevindjohnson.org/articles/sympalantir-rag-why-rag-security-requires-heterogeneous-compute.html, Mar. 2026.

[9] ——, "SymCognitive: AI-enabled financial ontology discovery," 2026, demonstration repository. 1,804 entities, 2,690 relationships from 37 financial firms.

[10] Palantir Technologies, "Foundry platform overview: Architecture," https://www.palantir.com/docs/foundry/platform-overview/architecture, see also: "Object Backend Overview," https://www.palantir.com/docs/foundry/object-backend/overview.

[11] ——, "AIP architecture overview," https://www.palantir.com/docs/foundry/architecture-center/aip-architecture.

[12] NVIDIA, "KAI scheduler," GitHub repository. https://github.com/NVIDIA/KAI-Scheduler, see Issues #848 (GPU over-allocation), #423 (no runtime GPU enforcement).

[13] ——, "GTC 2026 news," https://blogs.nvidia.com/blog/gtc-2026-news/, Mar. 2026, see also: "RTX AI and NemoClaw." Nemotron 3 Super 120B: 120B total parameters, 12B active per token, hybrid Mamba-Transformer backbone.

[14] S. Zhou, "Performance studies of dynamic load balancing in distributed systems," Ph.D. dissertation, Dept. of Electrical Engineering and Computer Sciences, Univ. of California, Berkeley, Oct. 1987, tech. Rep. UCB/CSD-87-376. Advisor: D. Ferrari. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1987/6223.html.

[15] L3Harris Technologies, "L3Harris and palantir demonstrate successful integration of new AI/ML capabilities," https://www.l3harris.com/newsroom/editorial/2024/10/l3harris-and-palantir-demonstrate-successful-integration-new-aiml, Oct. 2024.

[16] BrainChip Holdings Ltd., "Akida AKD1000 technical reference," https://brainchip.com/akida-neural-processor-soc/, including CNN2SNN conversion pipeline and MetaTF SDK.

[17] S. Minott, S. Siddiqui, and R. J. Haddad, "Benchmarking edge AI platforms: Performance analysis of NVIDIA jetson and raspberry pi 5 with coral TPU," in *Proc. IEEE SoutheastCon 2025*, 2025, pp. 1384–1389.
[18] Google Coral, "Performance and power consumption," https://developers.google.com/coral/guides/power.
[19] S. Kaczmarek, "Benchmarking the energy cost of assurance in neuromorphic edge robotics," *arXiv preprint arXiv:2603.13880*, Mar. 2026.
[20] IBM, "IBM storage scale (GPFS) documentation," https://www.ibm.com/docs/en/storage-scale, including extended attributes (xattrs), Information Lifecycle Management (ILM), Active File Management (AFM), and RDMA.
[21] ——, "IBM storage reference architecture with NVIDIA DGX A100 systems," https://images.nvidia.com/content/data-center/resources/IBM-ESS-3000-DGX-POD-RA-v13.pdf, 2023, see also: IBM, "IBM Storage Scale 6000: Powering the AI Factory," IBM Newsroom, 2025.
[22] W. Kwon *et al.*, "Efficient memory management for large language model serving with PagedAttention," in *Proc. SOSP '23*, 2023.
[23] K. D. Johnson, "SymVLLM: vLLM inference under symphony SOAM with 38-metric ELIM reporting," 2026, demonstration repository. 20 vLLM metrics + 18 GPU metrics per instance, sub-millisecond ELIM query latency, 10 ms total routing decision.
[24] Red Hat Developer, "llm-d: Kubernetes-native distributed inferencing," https://developers.redhat.com/articles/2025/05/20/llm-d-kubernetes-native-distributed-inferencing, May 2025.
[25] K. D. Johnson, "SymVLLM-Distributed: Distributed inference with scoring algorithms on symphony ELIM," 2026, demonstration repository. Four routing scorers implemented on ELIM, four-GPU consumer deployment, GPFS multi-tier KV cache architecture.
[26] ——, "SymLLMRouter: Semantic routing with KNN classification and neuromorphic integration," 2026, demonstration repository. 91.1% classification accuracy, 74–81% cost savings, 185 qps routing throughput, React Native mobile application.
[27] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models while reducing cost and improving performance," *arXiv preprint arXiv:2305.05176*, 2023, published in TMLR, 2024.
[28] I. Ong *et al.*, "RouteLLM: Learning to route LLMs with preference data," *arXiv preprint arXiv:2406.18665*, 2024.
[29] K. D. Johnson, "Unified training and inference orchestration for NVIDIA AI factories: A dual-scheduler architecture with LSF, symphony and GPFS," Technical Paper, Mar. 2026, companion paper.
[30] IBM, "IBM cloud for financial services," https://www.ibm.com/cloud/financial-services.
[31] ——, "IBM cloud compliance programs," https://www.ibm.com/products/cloud/compliance.
[32] Y. Liu *et al.*, "LMCache: An efficient KV cache layer for enterprise-scale LLM inference," *arXiv preprint arXiv:2510.09665*, Oct. 2025.