



Sokovan: An Accelerator-first Kubernetes-compatible AI Container Engine

Joongi Kim / Lablup Inc.

2025-11-05



Table of Contents

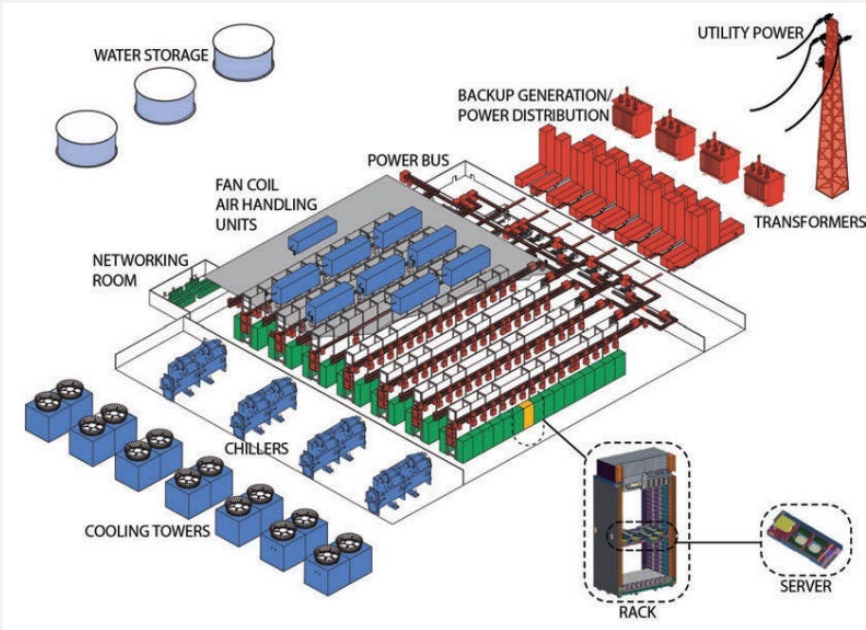
- AI Infra Challenges
- Is Kubernetes Doing Well for AI?
- Sokovan: Bringing Acceleration-native AI Containers to the Kubernetes World
- Future Directions



AI Infrastructure Challenges



AI Infrastructure Challenges



Interconnect Bottleneck

Internal delay limiting overall speed.



Cooling Infrastructure

Insufficient cooling restricting performance.



Utilization Efficiency

AI resources suffer underuse and imbalance.



Scheduler Scalability

Complex workloads need scalable scheduling.



Observability/Telemetry

Proper monitoring and diagnose is crucial.

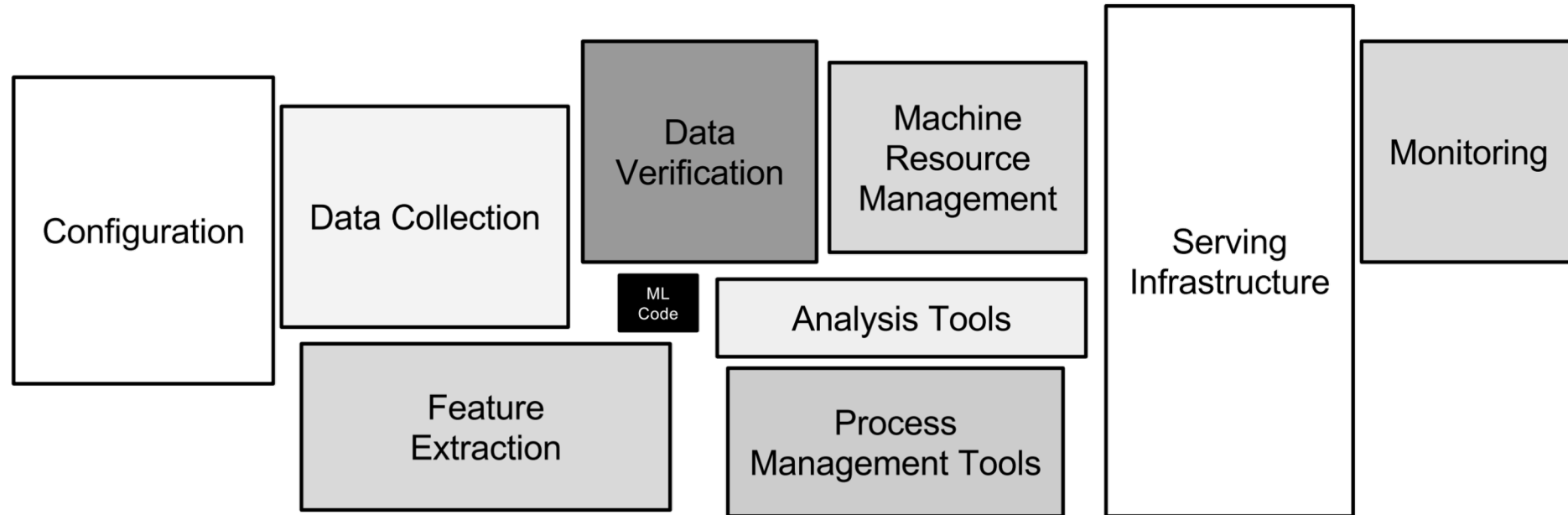


Data/Model Security

Threats evolve and systems grow complex.

[1] <https://www.construction-physics.com/p/how-to-build-an-ai-data-center>

AI Software Stack Challenges



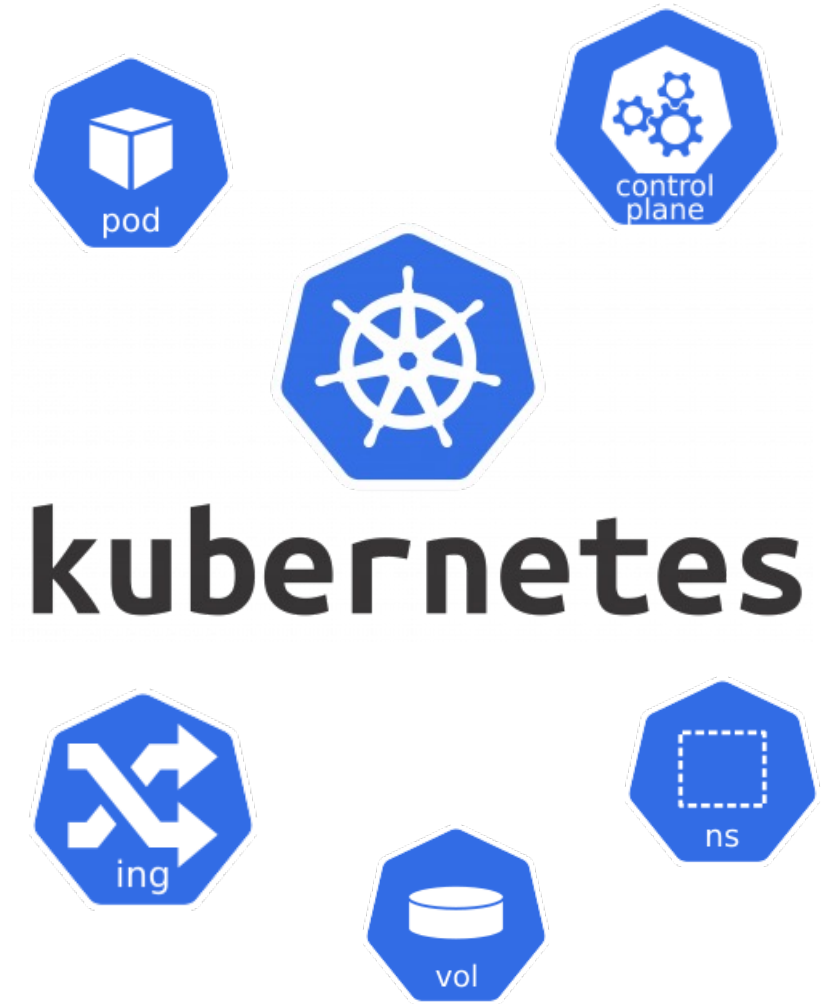
Courtesy of NeurIPS 2015 Paper "Hidden Technical Debt in Machine Learning Systems" (D. Sculley et al.)

Is Kubernetes Doing Well for AI?



Kubernetes

- Container Orchestration Engine
 - Reincarnation of 15-years of experience in Google Borg
 - July 2015: v1.0 Release / Donated to CNCF by Google
- Design Philosophy
 - Declarative Configuration
 - Controllers via Reconciliation loop
 - Immutable Containers
- Recent Advances regarding AI/ML Workloads (2023~)
 - DRA (Dynamic Resource Allocation)
 - In-Place Pod Resizing



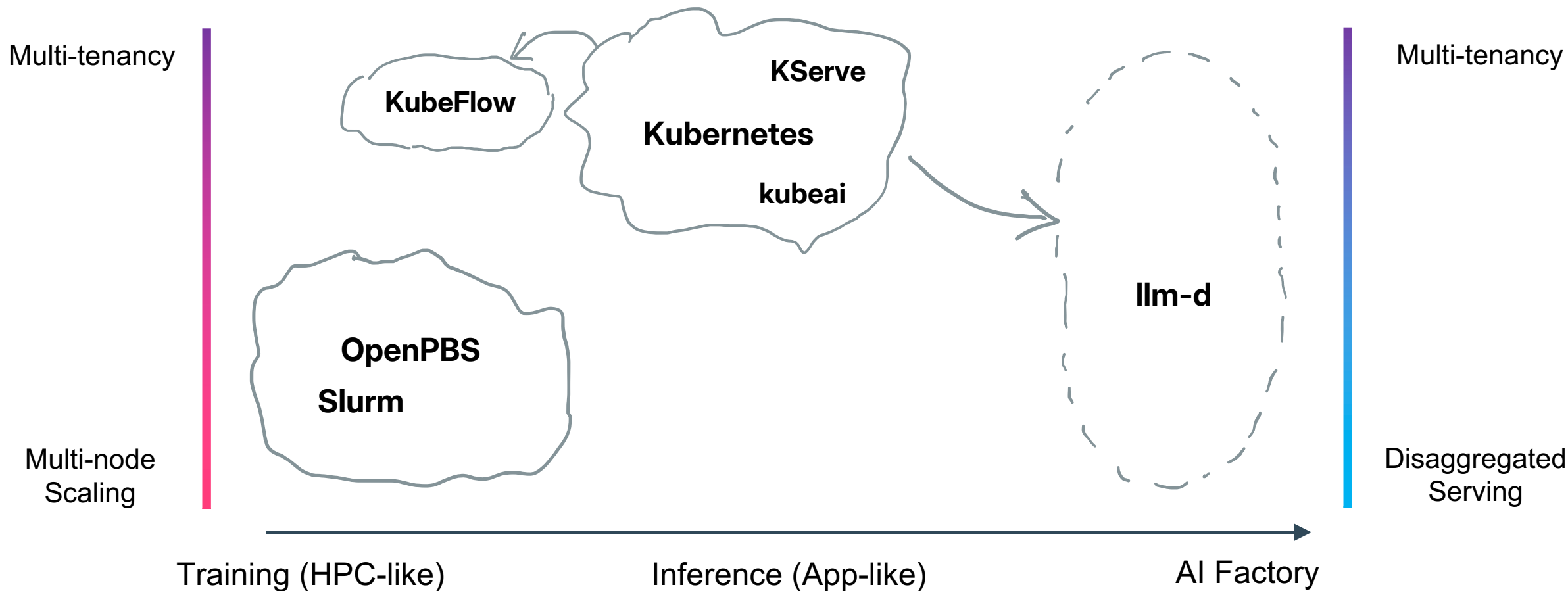
AI/HPC Workloads

- Key Characteristics & Requirements
 - Assumption: A job can consume all *visible* computing resources!
 - Multi-node scaling of individual jobs
 - High-speed east-west inter-node networking (RDMA & GPUDirect Storage)
 - Batch-oriented, stateful nature of jobs → Restarting & relocation is expensive!
 - Shared data volumes across jobs and pipeline stages (and nodes)
 - Extensive caching on local disks of individual nodes
- All these make it difficult on K8s...
 - <https://medium.com/better-programming/kubernetes-was-never-designed-for-batch-jobs-f59be376a338>

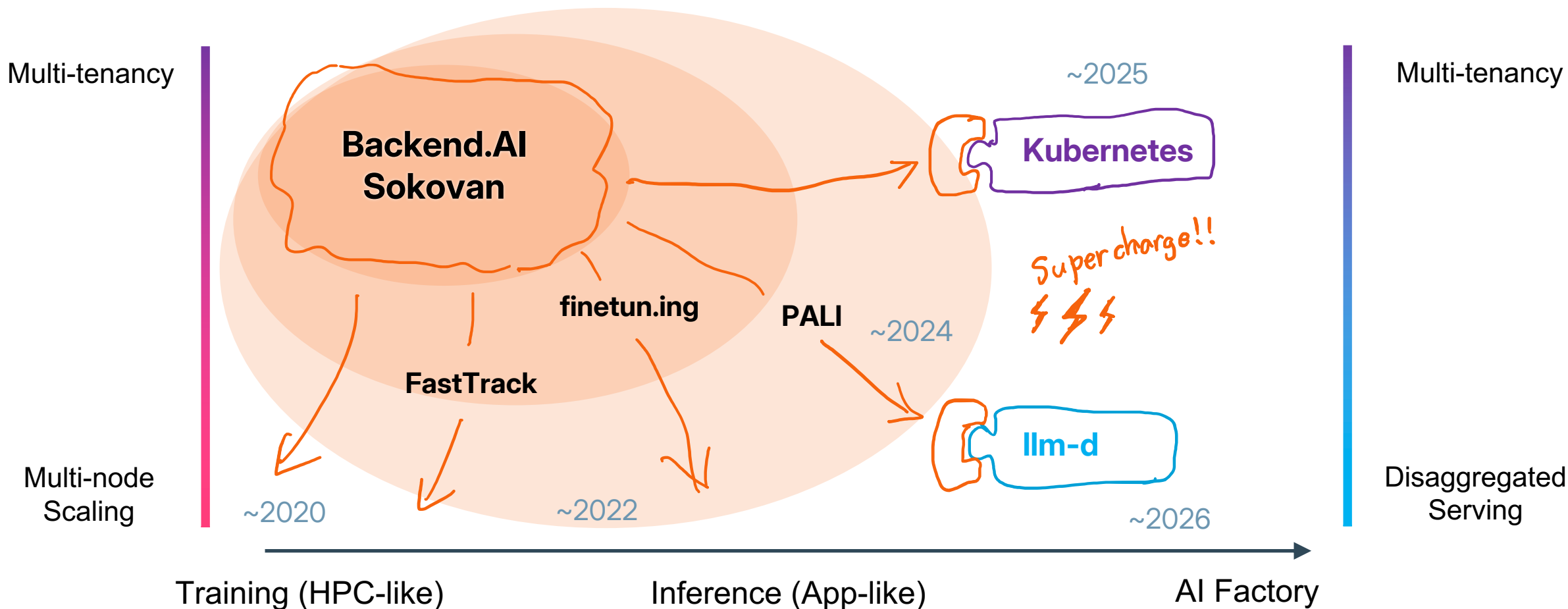
Containers, Kubernetes + AI & HPC

- Container is a *leaky* sandbox exposing many host information, while CPU/memory constraints are enforced by the Linux kernel (cgroup).
 - e.g., /proc/cpuinfo, sysfs, ...
 - lxcfs helps but sometimes crashes ML runtimes or frameworks.
- Accelerator integration & resource constraining relies on vendor-specific plugins.
 - e.g., nvidia-container-runtime
- Multi-node scaling is difficult with Pods.
 - Pod is a group of containers provisioned at a single host.
 - Multi-node jobs on K8s require *an additional coordination layer* to map pre-allocated pods to run partitions of the workload.

AI Workload Manager Landscape



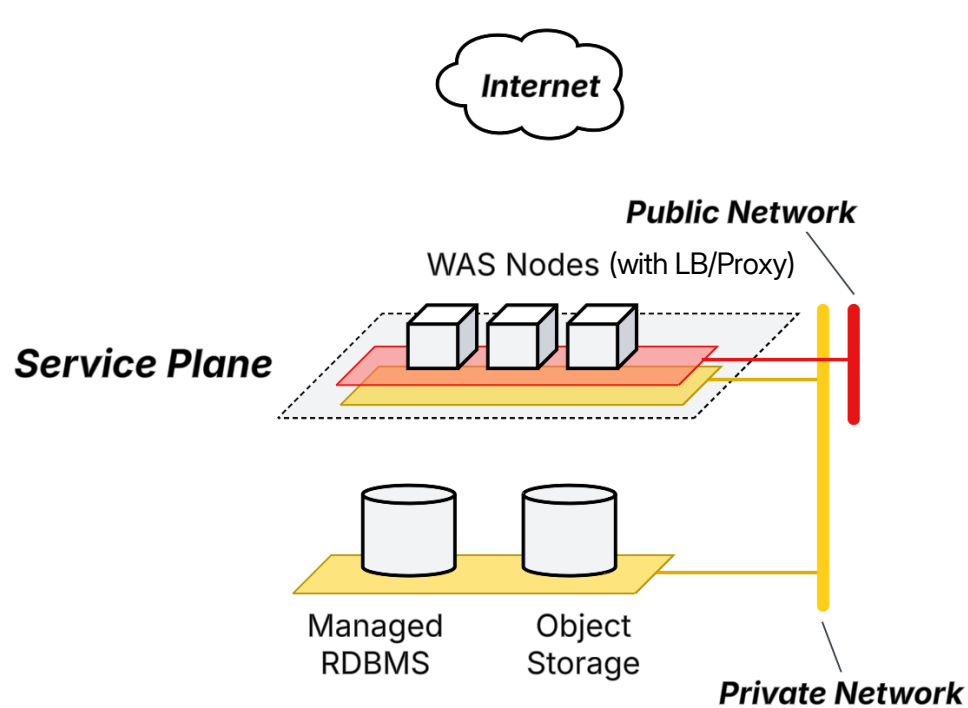
AI Workload Manager Landscape



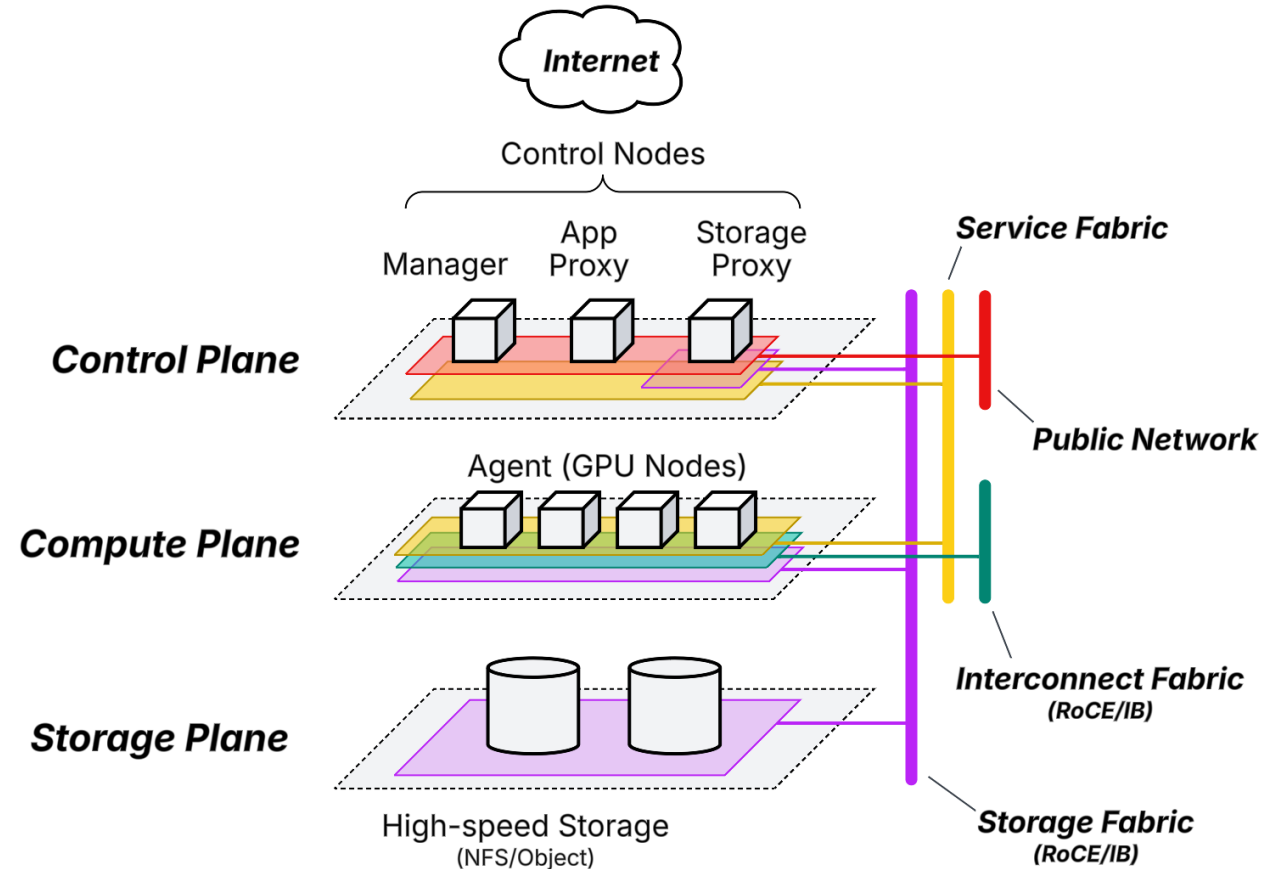
Sokovan: Bringing Acceleration-native AI Containers to the Kubernetes World



Complexity of AI Stack Deployment



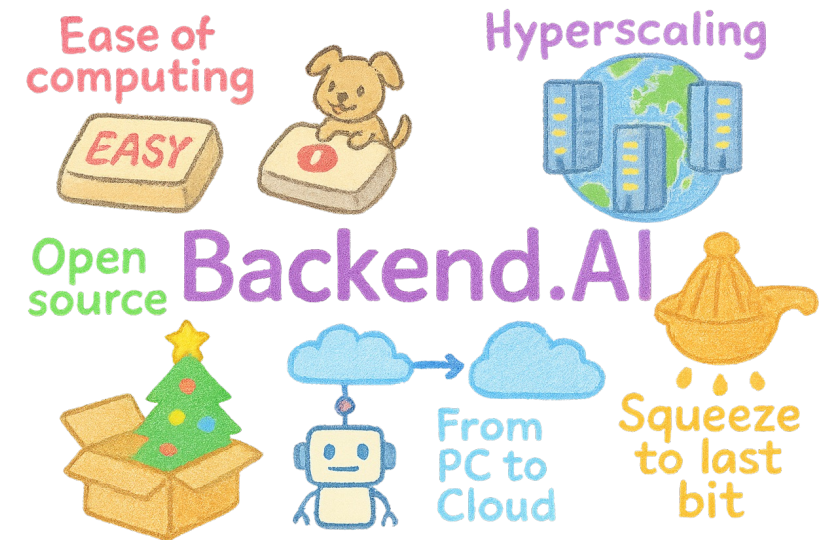
Conventional Web Service Stack



AI Service Stack (Native Sokovan)

Sokovan Container Engine as the core of Backend.AI

- Conceived in 2014
 - **Reproducible computing** environments for **multi-tenant** setups
 - **Acceleration-first**, hardware-aware sandboxing
 - Target: HPC workloads → AI
- Tech Stack & Design Choices
 - Docker Containers
 - Python asyncio + aiohttp
 - Resource **allocation** *instead of* resource **constraints**
- Advantages
 - **Multi-node job** scaling + multi-tenancy isolation
 - Proprietary GPU virtualization for containers (**fractional GPUs**)
 - **RDMA (GPUDirect Storage) integration** with selected storage vendors
 - One-stop solution to develop & deploy AI, packaged for **air-gapped** enterprise clusters



Why Kubernetes Integration?

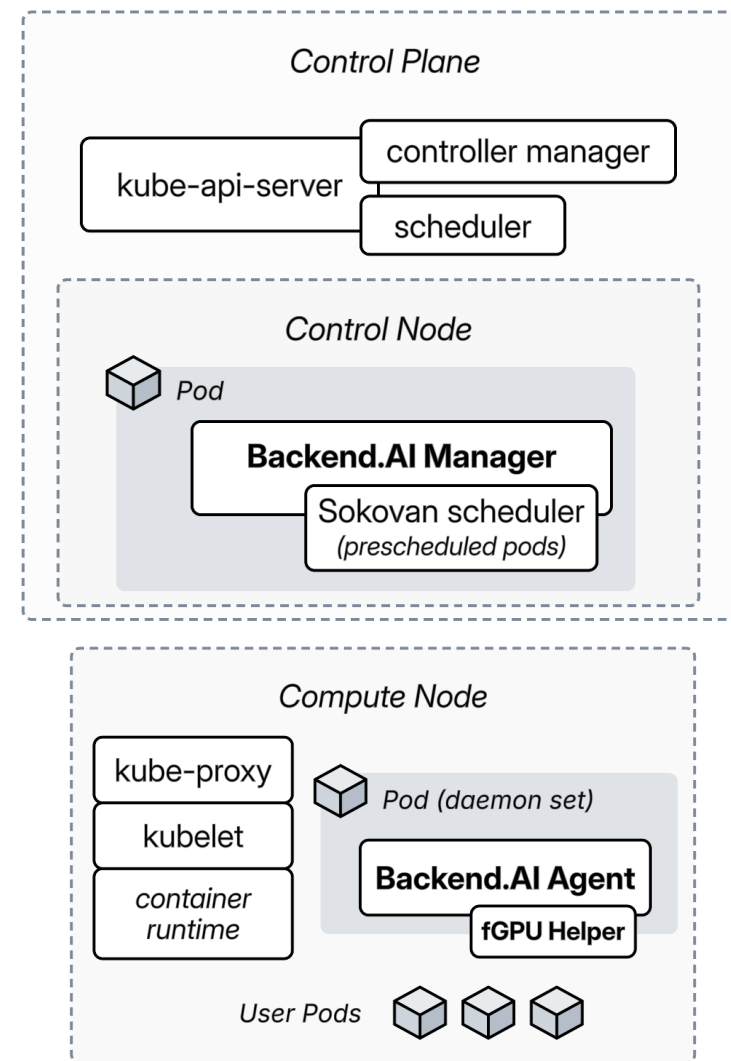
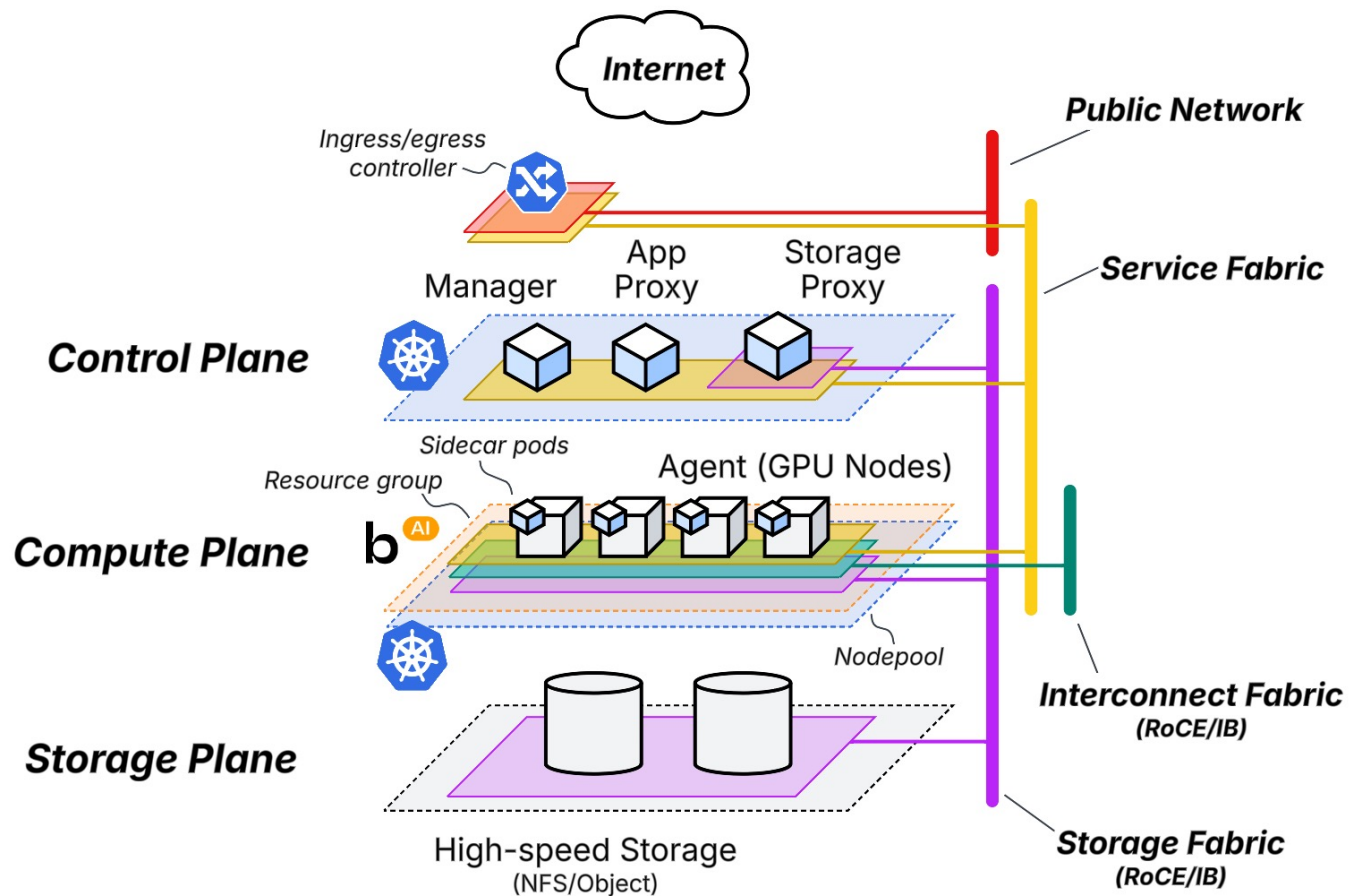
- Evolution of Enterprise IT Infrastructure
 - Previous: Collection of individual solutions and applications
 - K8s is a part of a solution or application.
 - Current: Unified management and monitoring on top of K8s
 - K8s is an operating system for hosting solutions and applications.
- Piggybacking on the Cloud-Native Ecosystem
 - Increasing number of participants and vendors providing K8s-based plugins and operators
 - e.g., Cilium to use eBPF-accelerated inter-container networking

Integration Principle & How

- Transparency & Observability to Existing K8s Tooling
 - Customer's own management tools should be able to observe Backend.AI-managed containers.
 - Backend.AI requires full control over GPU nodes and containers.
 - Namespacing and access control is essential.
- Performance & Hardware Acceleration as in Native Sokovan
 - We want the full performance and benefits of the Sokovan engine.
 - This is the challenging part.
- Entity Mapping

Sokovan	Kubernetes	Role
Resource Group	Node Pool	A collection of compute nodes sharing a same ACL
Kernel	Pod	A container running on a compute node
Session	(Group of Pods)	A bundle of containers representing a single computing job

Sokovan Stack on K8s (beta)

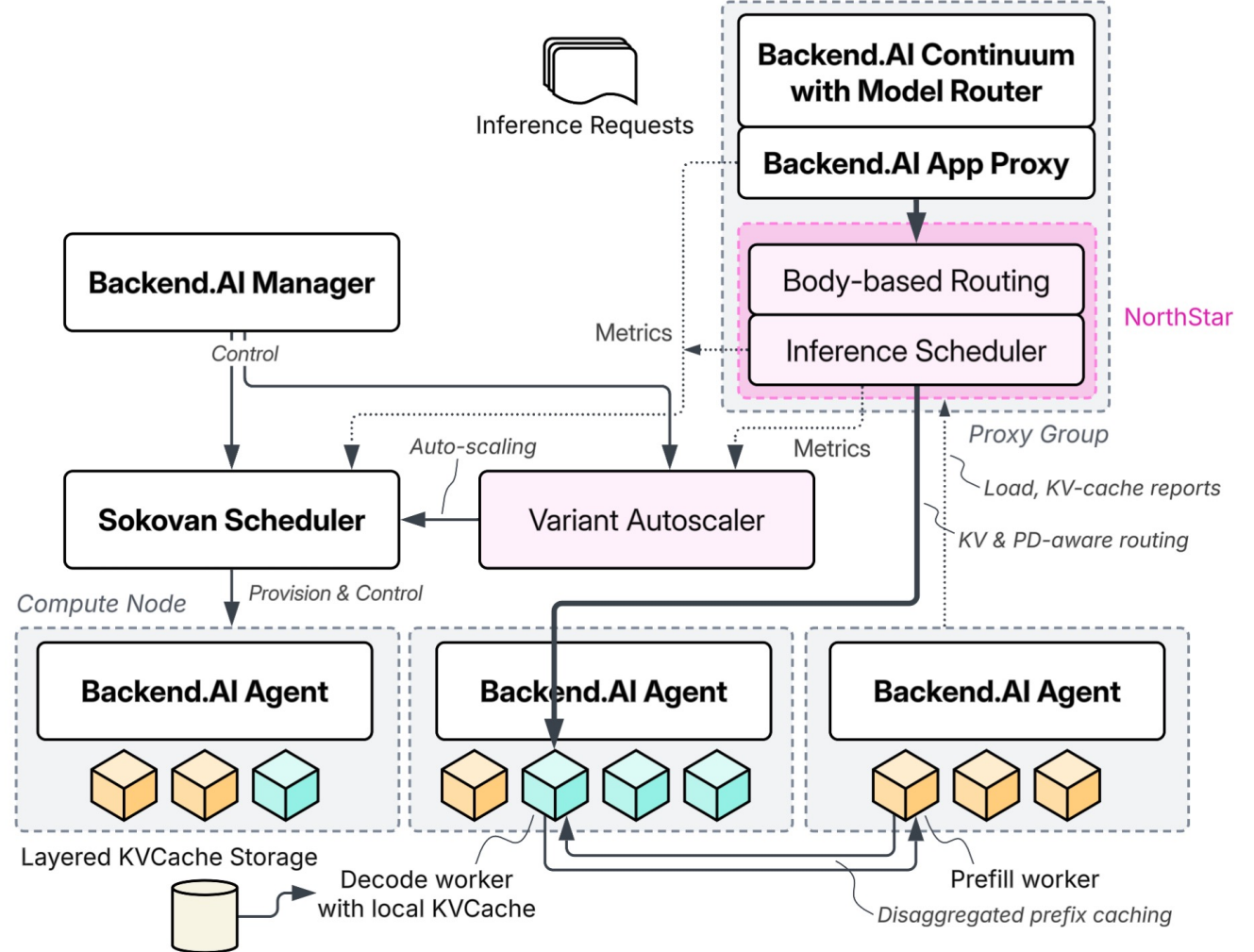


Future Directions



Future Directions

- Disaggregated Serving
 - Optimize GPU/NPU resource usage by splitting prefill and decode stages of transformer models to increase the request processing throughput under high concurrency
 - Requires more sophisticated coordination for KVCache sharing (e.g., NIXL)
 - llm-d is still in a very early stage! (launched 5 months ago)



Future Directions

- **CNI / CSI / CRI Integration**
 - Need to define the boundary and shared portion of the cloud-native stack
 - Goal: reuse the existing ecosystem as much as possible
- **Confidential Computing (with kubevirt and/or kata containers)**
 - Privacy-preserving Autonomous Agents
 - Cloud-hosted & self-hosted MCPs
- **App Hosting**
 - Deploying & serving AI-generated apps with cloud or hosted database backends
- **Enterprise Deployment**
 - Based on our experiences, there must be frictions with internal tooling, security, and policies!
 - Depending on the range of controllable scope, the difficulty may vary.



THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
KOREA

Sokovan: An Accelerator-first Kubernetes-compatible AI Container Engine

joongi@lablup.com

End of Talk: Thank you!

