Topology-Aware Scheduling for Large-Scale AI Workloads in Diverse Networks Clusters Using Volcano

Xiaodong Ye, Moore Threads

Yu Zhou, Moore Threads

KCD
GROWING CLOUD NATIVE TOGETHER
BEIJING

Content

Challenges in Large-Scale GPU Cluster Operations We built MTT KUAE! (10,000+ GPUs Cluster)

- Compute & Communication Efficiency
  - Massive parallelism with over 10,000 GPUs
  - Overlapping computation and communication
  - Optimized strategies: data, pipeline, tensor, and sequence parallelism
- Stability & Fault Tolerance
  - High risk of node/GPU/NIC failures during long-running jobs
  - Robust health checks, checkpointing, and quick recovery strategies
- Data Pipeline & I/O Bottlenecks
  - Avoiding redundant data reads
  - Asynchronous preprocessing and shared memory usage
- Network & Resource Scheduling
  - High-bandwidth network topology design and adaptive routing
  - Balancing large jobs vs. small jobs in a multi-tenant environment

System-Level (Cluster Management) Challenges

- Scheduling & Resource Allocation
  - Gang scheduling for multi-node, multi-GPU jobs
  - Handling diverse workloads (large jobs vs. small jobs)
- Health Monitoring & Fault Isolation
  - Periodic health checks for GPUs, PCIe, network, and storage
  - Automated job requeue and node remediation to prevent cascading failures
- IB/RoCE Network Optimization
  - Designing efficient network topology and communication patterns
  - Adaptive routing and congestion control to maintain low latency
- Scalability & Maintenance
  - Monitoring key metrics (MFU, ETTR, Goodput)
  - Ensuring cluster expansion does not compromise reliability

Category	Reason	Num	GPU Demand		Time to Failure (mins)		GPU Time (mins)		Time to Restart (mins)			Cluster
			Average	Median	Average	Median	Average	Total%	Average	Median	TR/TF%	
Infrastructure	NVLink Error	54	800	896	868.1	155.3	585683	30.25%	95.6	0.2	11.02%	S, K
	CUDA Error	21	847	1024	923.2	586.0	785099	15.77%	78.3	2.0	8.48%	S, K
	Node Failure	16	712	768	1288.8	535.8	934394	14.30%	102.8	21.5	7.98%	S
	ECC Error	12	680	512	1303.4	1192.3	958404	11.00%	2.8	1.8	0.21%	S, K
	Network Error	12	758	768	549.6	310.1	394821	4.53%	592.1	7.4	107.74%	S, K
	Connection Error	147	29	1	51.9	0.5	24492	3.44%	0.8	0.0	1.51%	S, K
	S3 Storage Error	10	422	256	2317.8	202.2	222151	2.12%	6.2	0.2	0.27%	S
	NCCL Timeout Error	6	596	512	159.7	48.1	86856	0.50%	66.7	43.6	41.78%	K
	NCCL Remote Error	3	1152	1024	50.5	22.6	52419	0.15%	0.0	0.7	0.09%	K
Framework	Dataloader Killed	6	445	508	1580.6	961.4	764170	4.38%	115.1	0.9	7.28%	K
	Attribute Error	67	228	8	67.8	1.2	60914	3.90%	2.4	0.0	3.58%	S, K
	Out of Memory Error	14	572	640	323.8	14.5	245278	3.28%	122.7	1.2	37.89%	S, K
	Runtime Error	65	441	352	66.4	3.9	27667	1.72%	10.9	1.5	16.41%	S, K
	Assertion Error	105	413	256	41.7	3.0	12315	1.24%	185.9	1.6	445.87%	S, K
	Value Error	33	387	256	9.9	3.7	5049	0.16%	27.4	0.6	276.74%	S, K
	Zero Division Error	5	499	256	14.5	15.6	5363	0.03%	2.5	1.1	17.31%	S, K
	Model Loading Error	104	8	8	2.6	2.6	20	0.00%	0.0	0.0	0.00%	K
	Dataset Loading Error	5	1	1	1.6	1.6	1	0.00%	0.0	0.0	0.00%	K
Script	File Not Found Error	568	21	1	14.2	0.4	5210	2.83%	0.4	0.0	2.58%	S, K
	OS Error	266	8	1	9.6	0.8	1098	0.28%	0.3	0.0	3.17%	S, K
	Type Error	620	18	4	0.9	0.3	97	0.06%	0.2	0.0	28.27%	S, K
	Name Error	18	247	24	3.2	0.5	947	0.02%	2.9	2.4	90.92%	S, K
	Permission Error	7	438	512	4.3	0.8	2131	0.01%	2.4	2.2	56.38%	S
	Import Error	111	93	8	1.1	0.4	74	0.01%	0.7	0.0	63.68%	S, K
	Key Error	260	7	0	3.0	1.6	55	0.01%	0.1	0.0	2.10%	S, K
	Syntax Error	10	391	384	0.7	0.6	348	0.00%	1.7	1.7	261.73%	S, K
	Argument Error	3	344	512	0.7	0.7	288	0.00%	2.7	0.7	408.47%	S
	Called Process Error	4	256	256	0.2	0.2	52	0.00%	11.7	10.9	5714.29%	S
	Index Error	23	6	1	1.6	0.9	21	0.00%	0.8	0.0	49.73%	S, K

Table 3: Job failure statistics. It is sorted based on **Total%** (i.e., the percentage of GPU time summation in different categories). **Num**: Number of Occurrence. **TF**: Time to Failure. **TR**: Time to Restart (i.e., Restart Timestamp − Failure Timestamp). **GPU Time**: TF×GPU Demand. **S/K**: Occurrence of errors in `Seren`/`Kalos` respectively.

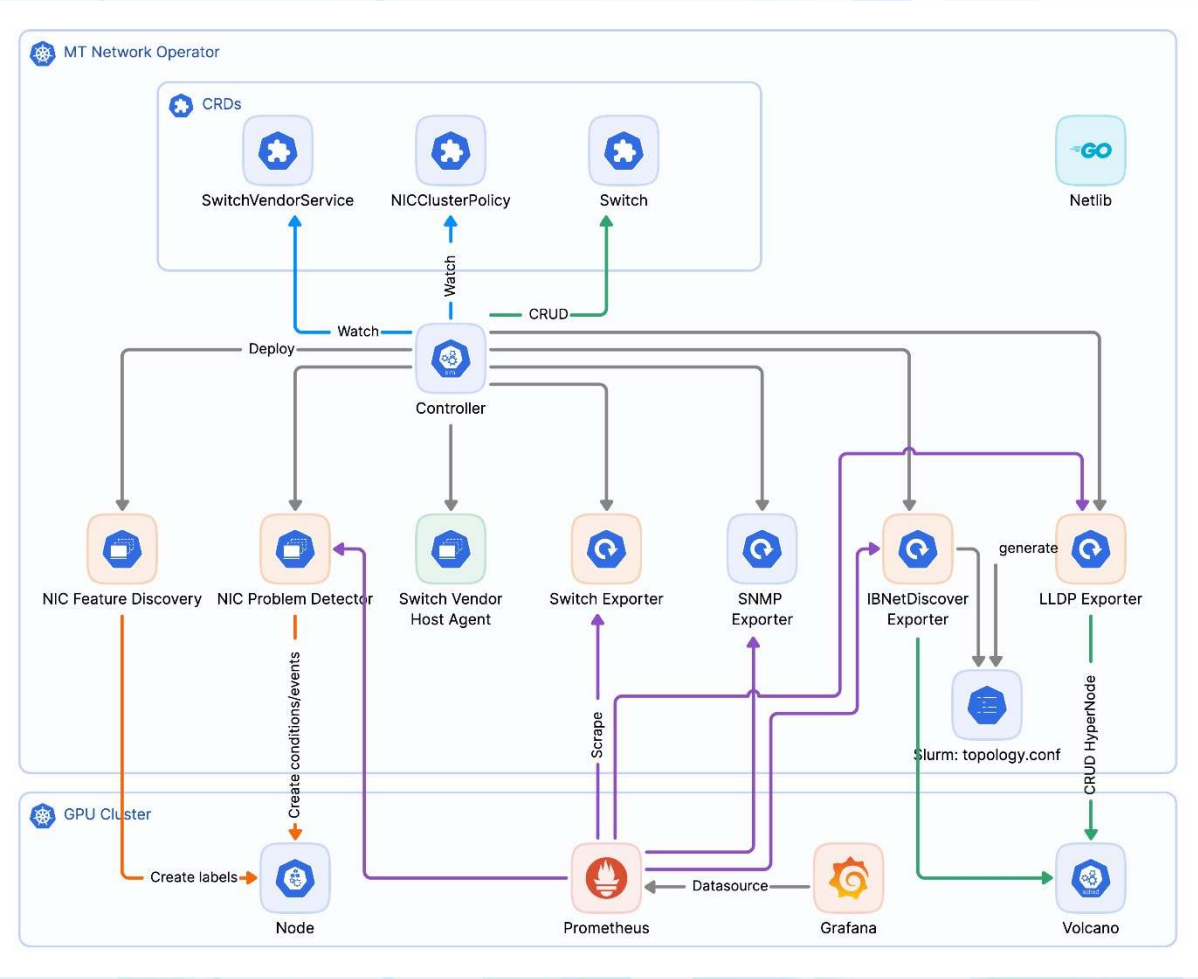
* From meta paper

Business-Level (Training/Inference) Challenges

- Training Efficiency & Model Convergence
  - Coordinating large-scale parallelism without compromising stability
  - Mitigating overhead from fault recovery and checkpointing
- Inference Service Performance
  - Real-time low-latency inference in distributed settings
  - Balancing training and inference resource needs in shared clusters
- Cost Efficiency & Energy Management
  - High operational and energy costs in large-scale clusters
  - Maximizing GPU utilization while managing expenses
- Business Continuity & Disaster Recovery
  - Long-duration training runs with minimal interruptions
  - Robust recovery mechanisms to ensure service availability

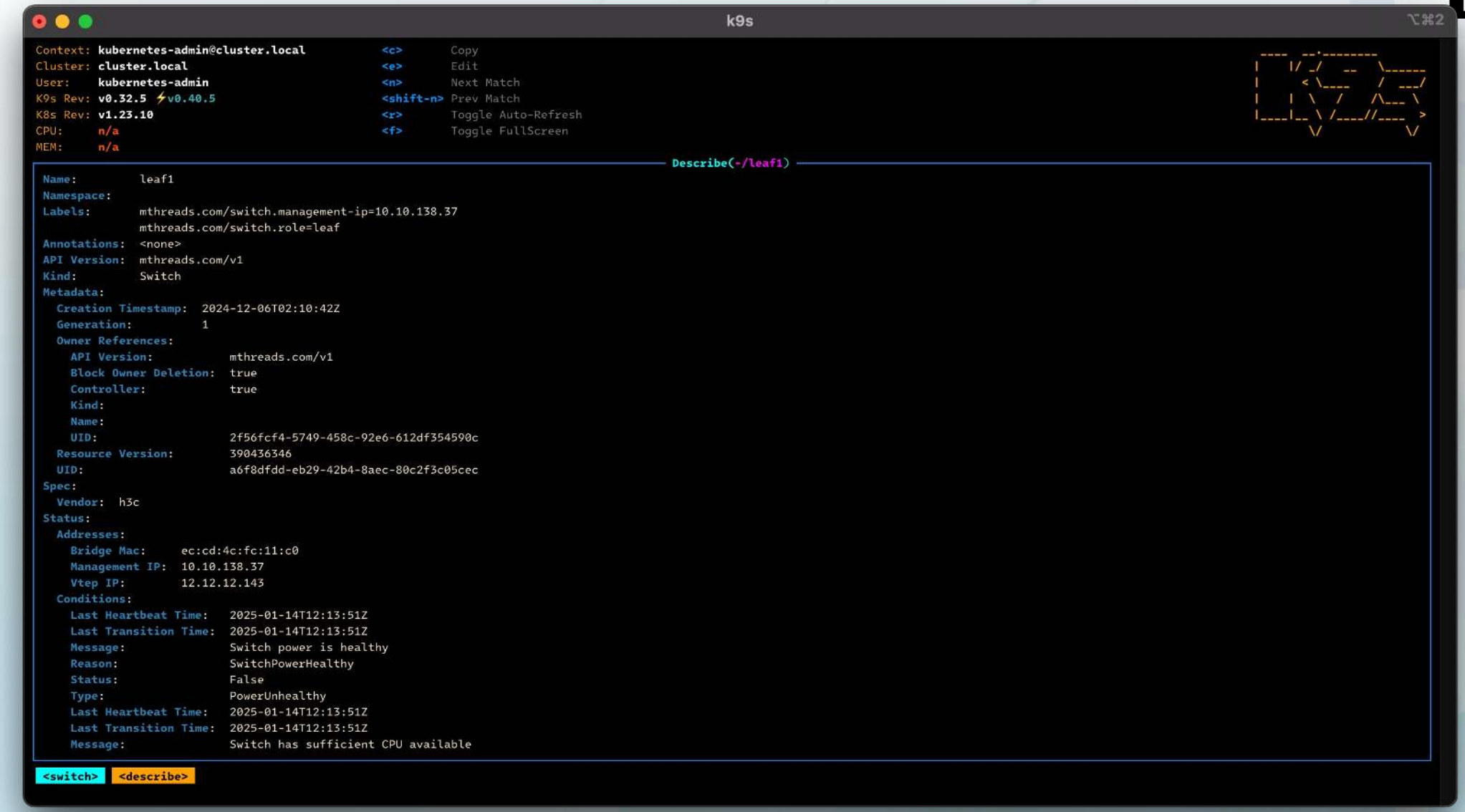
Network Monitoring, Fault Detection & Fault Labeling Initiatives

- Comprehensive Network Observability
  - Real-time monitoring and logging for host NICs, switches, and IB/RoCE topologies
  - Automated topology discovery to map network interconnections
- Automated Fault Detection & Labeling
  - Early identification of anomalies and failures in network components
  - Precise fault labeling to streamline root cause analysis and remediation
- mt-network-operator
  - Seamlessly integrates with existing cluster management systems
  - Enables proactive network health tracking and fault management
- Benefits & Future Directions
  - Reduced downtime and improved reliability through early intervention
  - Enhanced network performance and resource utilization

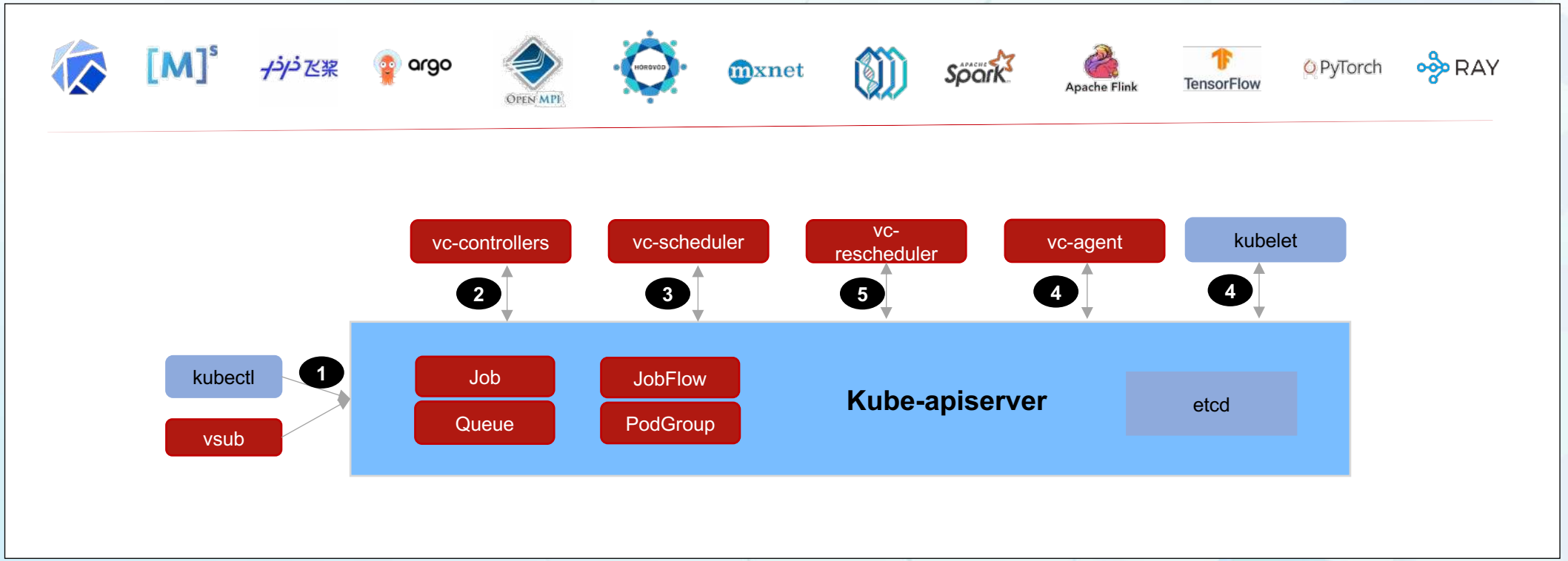


* mt-network-operator

mthreads.com/v1/Switch

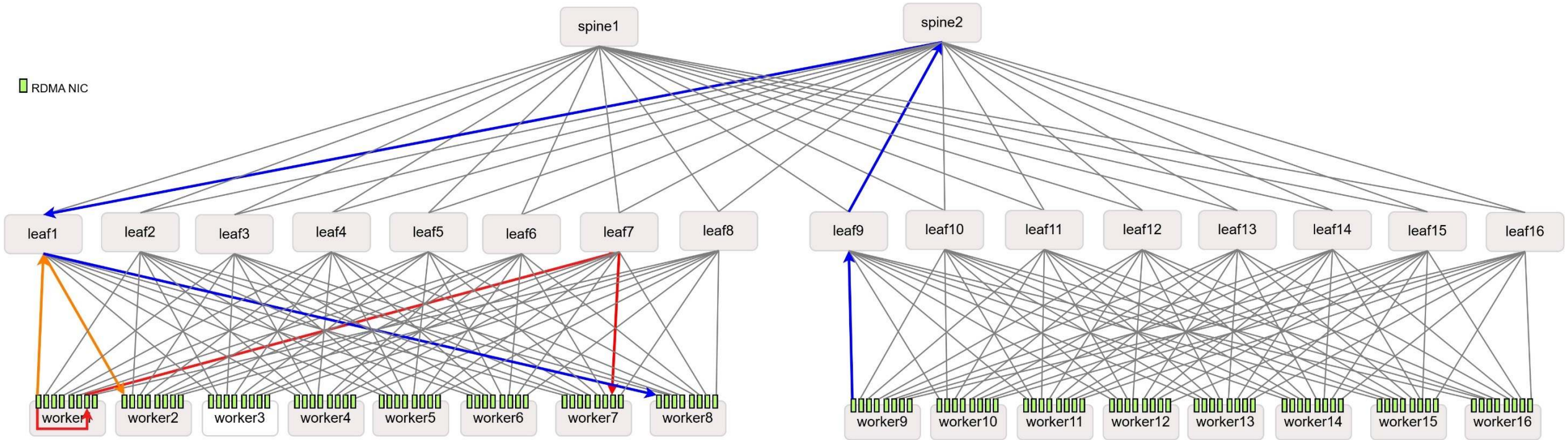


Volcano overview



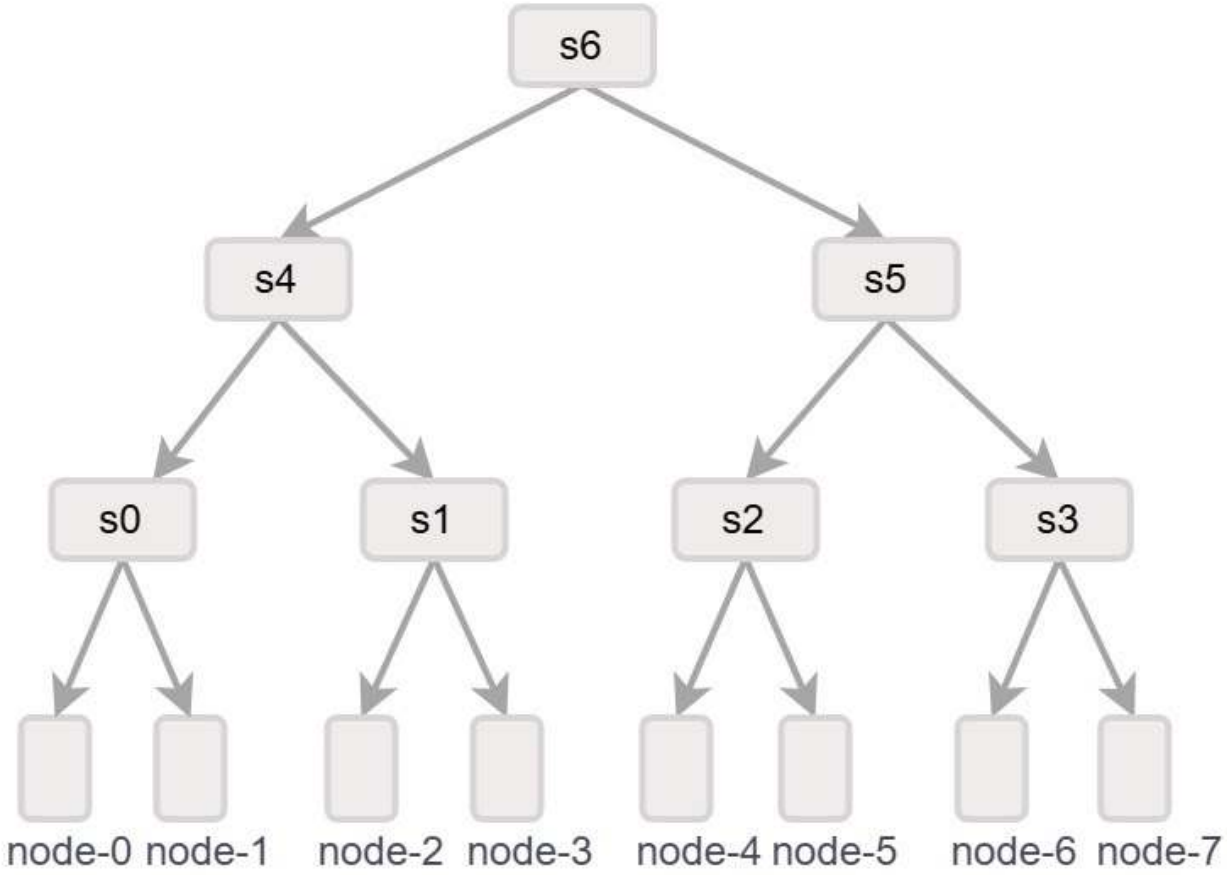
Network Design

- Fat-Tree
  - High Bandwidth and Scalability
  - Optimized for Distributed Training
  - Fault Tolerance and Reliability



HyperNode -- Network Topology API

- **Leaf HyperNodes** (s0, s1, s2, s3): The child node type is the real nodes in the cluster.
- **Non-leaf HyperNodes** (s4, s5, s6): The child node type is other HyperNodes.

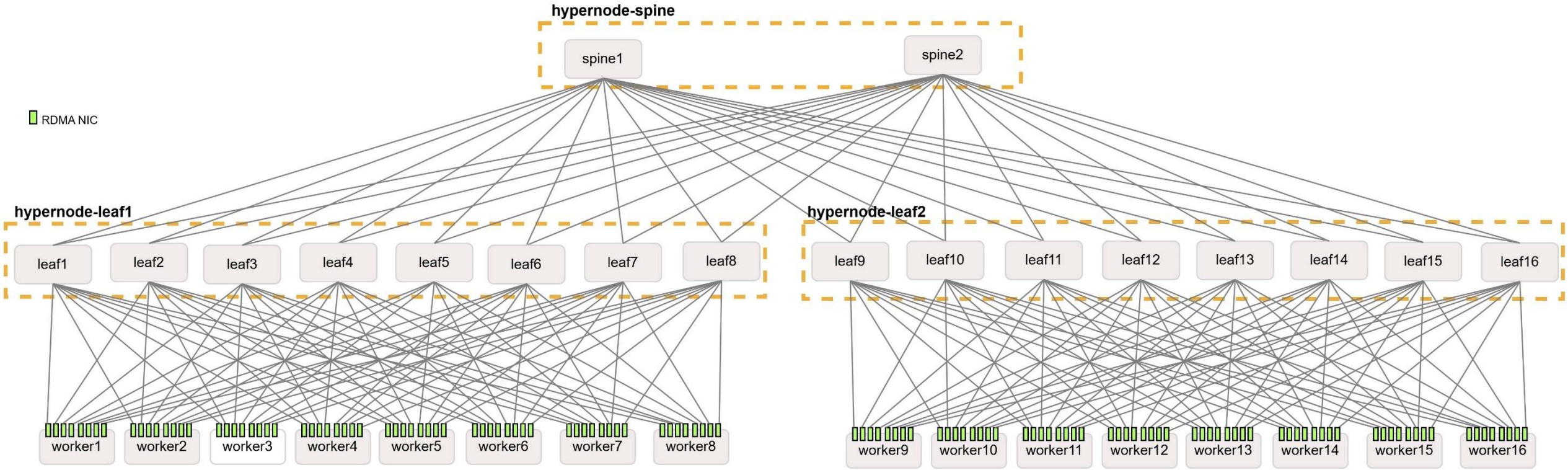



```yaml
apiVersion: topology.volcano.sh/v1alpha1
kind: HyperNode
metadata:
  name: s0
spec:
  tier: 1 # s0 is at tier1
  members:
  - type: Node
    selector:
      exactMatch:
        name: "node-0"
  - type: Node
    selector:
      exactMatch:
        name: "node-1"
```

Build HyperNode Structure

HyperNodes are structured as follows based on the actual conditions:**"A HyperNode represents a network topology performance domain**, typically mapped to a switch or tor".

In fat-tree network architecture, the HyperNode represents **a connectivity domain** rather than the concept of a switch.



Volcano Configuration

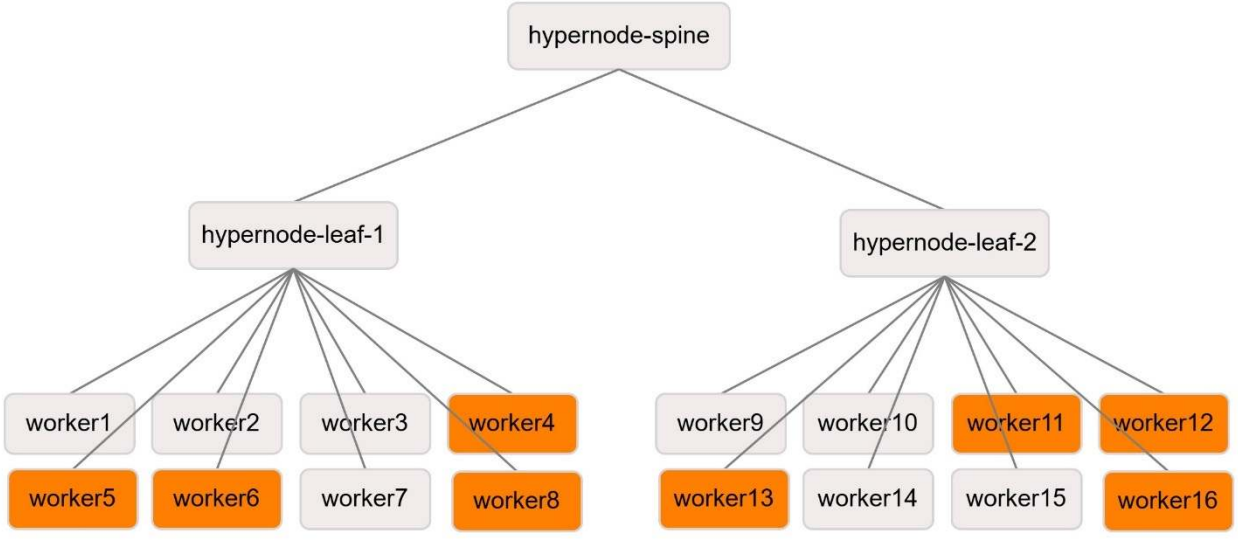
- To reduce resource fragmentation:
  - Enable binpack plugin
  - Set an appropriate weight

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: volcano-scheduler-configmap
  namespace: volcano-system
data:
  volcano-scheduler.conf: |
    actions: "enqueue, allocate, backfill"
    tiers:
    - plugins:
      - name: priority
      - name: gang
    - plugins:
      - name: predicates
      - name: proportion
      - name: nodeorder
+     - name: binpack
+       arguments:
+         binpack.weight: 10
```

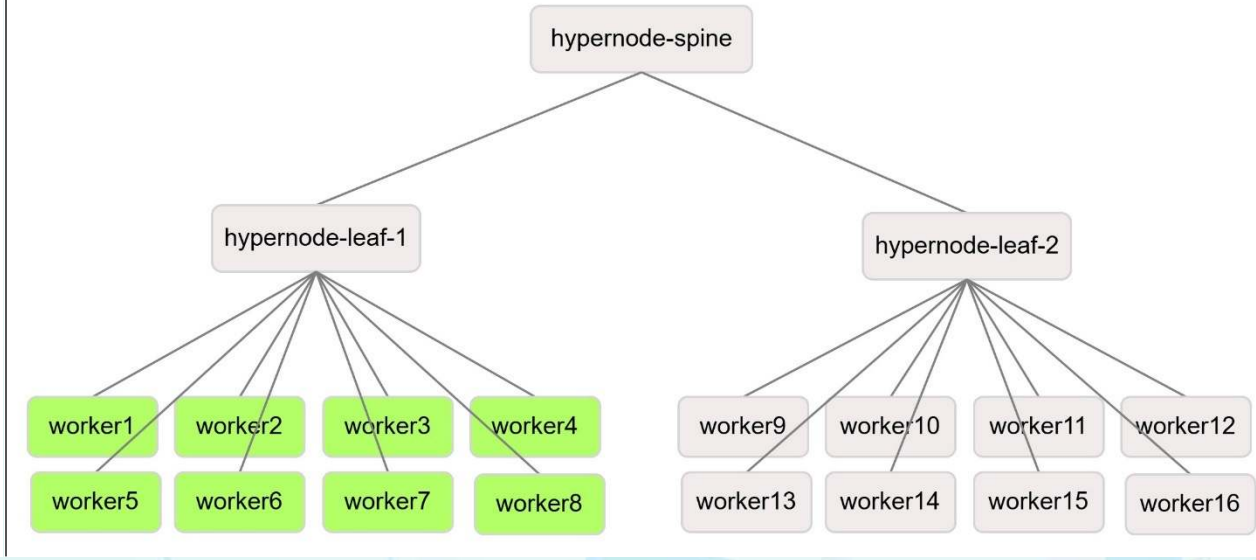
```
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: test-job
spec:
  minAvailable: 8
  schedulerName: volcano
  networkTopology: # Set network topology constraints
    mode: hard
    highestTierAllowed: 1
  queue: default
  tasks:
    - replicas: 8
      name: "pod"
      template:
        spec:
          containers:
            - command: ["sleep"]
              args: ["infinity"]
              image: sh-harbor.mthreads.com/cloud/ubuntu:20.04
              imagePullPolicy: IfNotPresent
              name: job
              resources:
                limits:
                  nvidia.com/gpu: 8
                  openshift.io/roce: 8
          restartPolicy: OnFailure
```

Schedule Results

- Normal Schedule



- Scheduling With Network Topology Aware



Demo


(base) root@yuzhou-System-Product-Name:~/go/src/github.com/yeahdongcn/kcd-beijing-2025/scripts#

Topology Discovery Mechanism

- InfiniBand Networker
  - IBNetDiscover Exporter
- RoCE(RDMA over Converged Ethernet) Network
  - LLDP Exporter

IBNetDiscover Exporter

- **IB Network** Topology Generator
- Discovery method:
  - ibnetdiscover: part of the infiniband-diags RPM package
  - slurmibtopology.sh: create a Slurm topology.conf file to get the correct node and switch Infiniband

```
###################################################################
# Slurm's network topology configuration file for use with the
# topology/tree plugin
###################################################################
SwitchName=s0 Nodes=dev[0-5]
SwitchName=s1 Nodes=dev[6-11]
SwitchName=s2 Nodes=dev[12-17]
SwitchName=s3 Switches=s[0-2]
```

- Topology Output: topology.conf and HyperNode
- **Topology**: A Go-based tool for scheduling nodes according to topology.conf

LLDP Exporter

- **RoCE Network** Topology Generator
- Discovery method:
  - Link Layer Discovery Protocol (LLDP)
  - Switch vendor topology API
- Topology Output: topology.conf and HyperNode

HyperNode Status

- **Node nic-related conditions**:

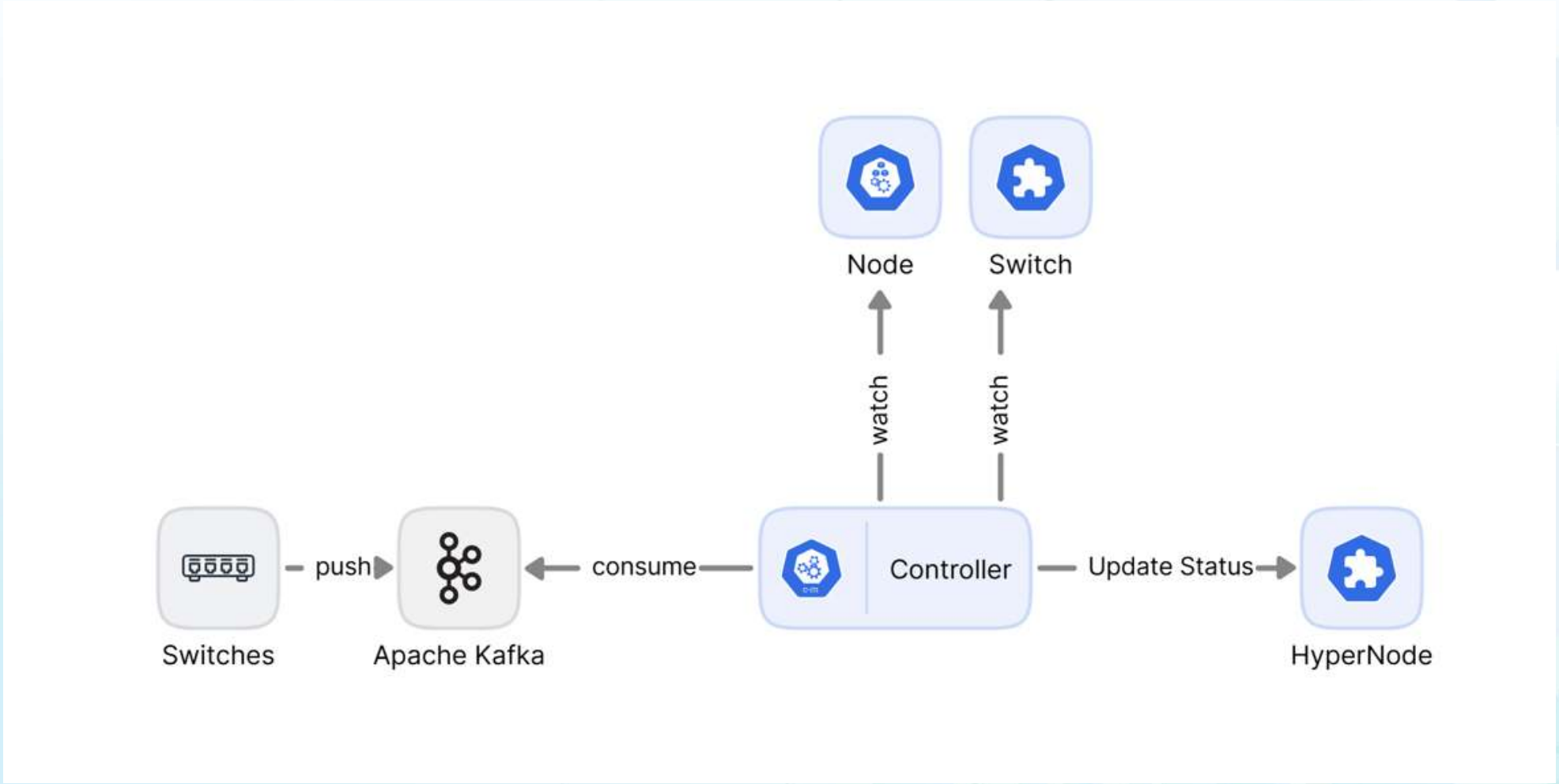
```
Conditions:
  Type                    Status    Reason          Message
  ----                    ------    ------          -------
  NicUnhealthy_ens2np0    False     NicIsHealthy    Nic on the node is functioning properly
  NicUnhealthy_ens3np0    True      NicIsHighBER    Raw Physical BER is greater than threshold
```

- **Switch conditions**

```
Conditions:
  Last Heartbeat Time:    2025-01-14T12:13:51Z
  Last Transition Time:   2025-01-14T12:13:51Z
  Message:                Switch fan is healthy
  Reason:                 SwitchFanHealthy
  Status:                 False
  Type:                   FanUnhealthy
  Last Heartbeat Time:    2025-01-14T12:13:51Z
  Last Transition Time:   2025-01-14T12:13:51Z
  Message:                Switch power is healthy
  Reason:                 SwitchPowerHealthy
  Status:                 False
  Type:                   PowerUnhealthy
  Last Heartbeat Time:    2025-01-14T12:13:51Z
  Last Transition Time:   2025-01-14T12:13:51Z
  Message:                Switch has sufficient CPU available
  Reason:                 SwitchHasSufficientCPU
  Status:                 False
  Type:                   CPUPressure
  Last Heartbeat Time:    2025-01-14T12:13:51Z
  Last Transition Time:   2025-01-14T12:13:51Z
  Message:                Switch has sufficient memory available
```

```
status:
  conditions:
  - lastTransitionTime: "2025-01-20T07:58:30Z"
    message: Nic on the node is functioning properly
    reason: NicIsHealthy
    status: "False"
    type: NicUnhealthy_ens2np0_leaf1_worker-30
  - lastTransitionTime: "2025-02-10T10:40:46Z"
    message: Raw Physical BER is greater than threshold
    reason: NicIsHighBER
    status: "True"
    type: NicUnhealthy_ens2np0_leaf1_worker-31
  - lastTransitionTime: "2025-02-10T10:40:46Z"
    message: Raw Physical BER is greater than threshold
    reason: NicIsHighBER
    status: "True"
    type: NicUnhealthy_ens3np0_leaf2_worker-31
  - lastTransitionTime: "2025-03-03T11:52:02Z"
    message: Device network is healthy
    reason: SwitchNetworkIsHealthy
    status: "False"
    type: SwitchNetworkFaults_leaf1
  - lastTransitionTime: "2025-03-03T11:52:02Z"
    message: Device network is healthy
    reason: SwitchNetworkIsHealthy
    status: "False"
    type: SwitchNetworkFaults_leaf2
  - lastTransitionTime: "2025-03-03T11:52:02Z"
    message: The device is healthy
    reason: SwitchSystemIsHealthy
    status: "False"
    type: SystemFaults_leaf1
```

HyperNode Controller



HyperNode Enhancement

- Current problems when use HyperNode:
  - There is **no existing mechanism to indicate the health status of individual nodes**. This lack of granularity prevents the scheduler from accurately identifying and handling unhealthy nodes.
  - **Standard Switch Condition Type Not Yet Defined**.
- What we are doing can be referred to in this pull request: volcano-sh/apis#155:
  - **Introduce a new field under Status**, such as **UnhealthyNodes**, to explicitly list nodes that are currently unschedulable under the given HyperNode.
  - **Define two common switch condition types** to standardize switch health reporting:
    - **SystemFailure**: Indicates a system-level issue on the switch or tor, such as CPU or memory overload, power failure, fan malfunction, or other critical system faults.
    - **NetworkUnavailable**: Indicates a network-related issue on the switch or tor, such as abnormal link status, interface failures, or other network disruptions.

Future Work

- Work with the volcano community to improve network topology scheduling functions.
    - Adapt lldp-exporter to HyperNode provider, refer https://github.com/volcano-sh/volcano/pull/4014

Related links

- https://volcano.sh/en/docs/network_topology_aware_scheduling/
- https://github.com/volcano-sh/apis/pull/155
- https://github.com/yeahdongcn/kcd-beijing-2025
- https://github.com/Mellanox/network-operator
- https://github.com/volcano-sh/volcano
- https://github.com/volcano-sh/apis
- https://github.com/volcano-sh/apis/pull/155
- https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
- https://github.com/operator-framework/operator-sdk
- https://github.com/volcano-sh/volcano/pull/4014
- https://github.com/yeahdongcn/topology
- https://slurm.schedmd.com/topology.conf.html
- https://github.com/OleHolmNielsen/Slurm_tools/tree/master/slurmibtopology

Thanks.

摩尔线程公众号

摩尔线程官网

摩尔学院官网