

RDMA 容器网络下的大规模AI训练探索

蓝维洲 DaoCloud

董卫国 中国移动云能力中心



Content 目录

01 Background

02 Container Network Requirements

03 Spiderpool





Part 01

Background



Background

Large models has become an important foundation for the development of artificial intelligence.

Training is difficult because training large models requires a very large amount of computation, so introducing multi machine and multi card distributed training is necessary.

A single training of a large model may take several tens of days. Solving the network communication problem during AI training can improve the success rate of training and reduce training time.



Distributed Training

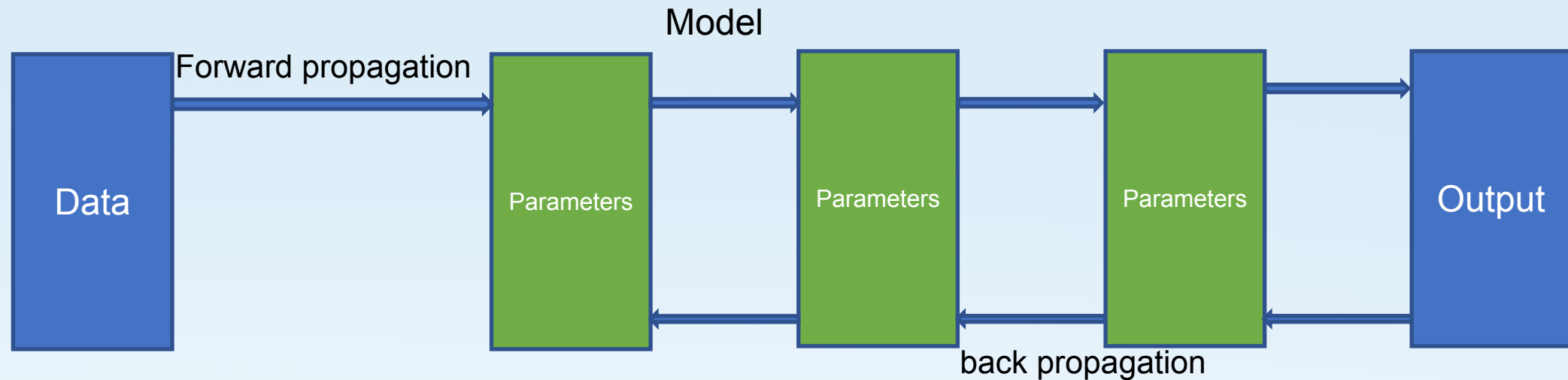
Horizontal expansion improves computing power and reduces model training time. Computational power can be roughly estimated using the following formula:

Computational power \propto Single device Computational power X Device number X Resource utilization rate

The computing speed of a single device is mainly determined by a single device (such as a GPGPU). In distributed training scenarios, the more computing devices there are, the higher the computing power available. However, due to the impact of communication efficiency, an increase in the number of computing devices may lead to a decrease in device utilization, resulting in actual usage of computing power not meeting expectations.

Model Training

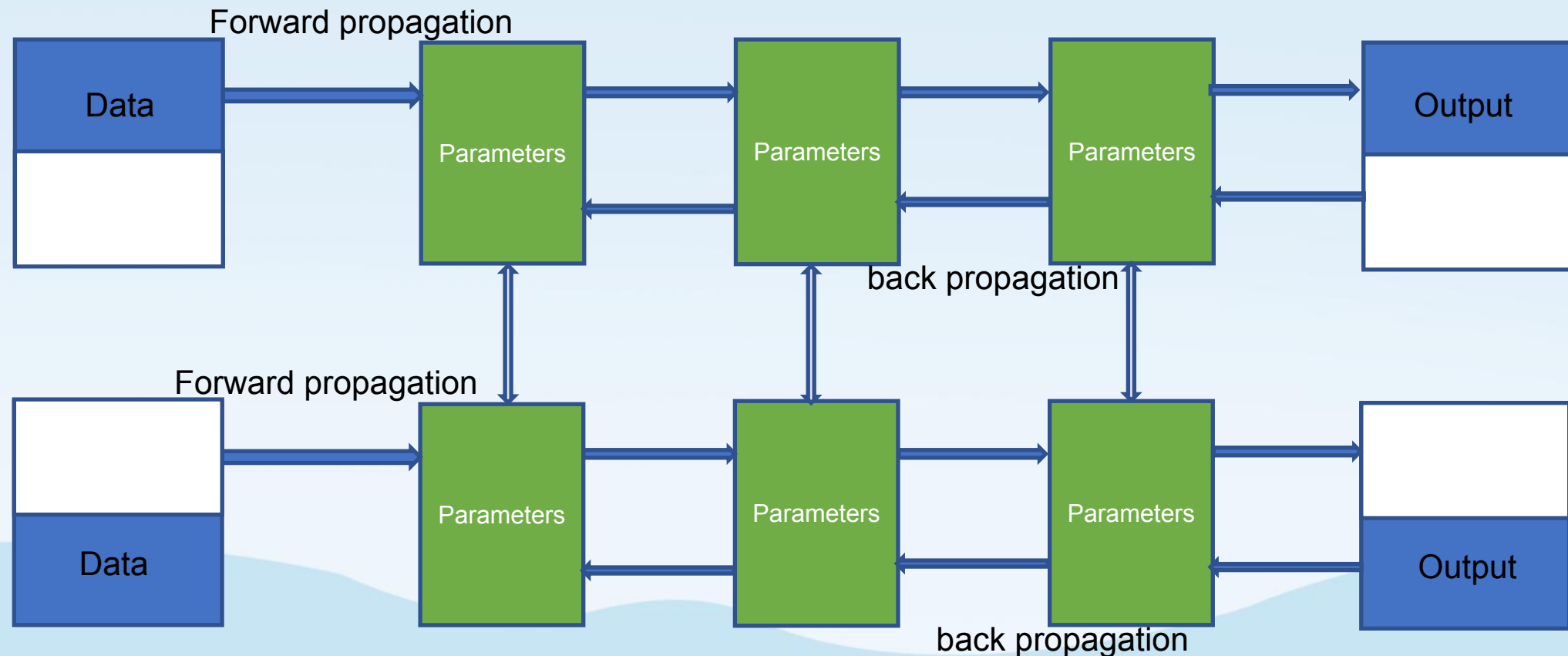
The model training process is the process of updating model parameters using optimization algorithms based on data and loss functions



Distributed training: data parallelism, model parallelism

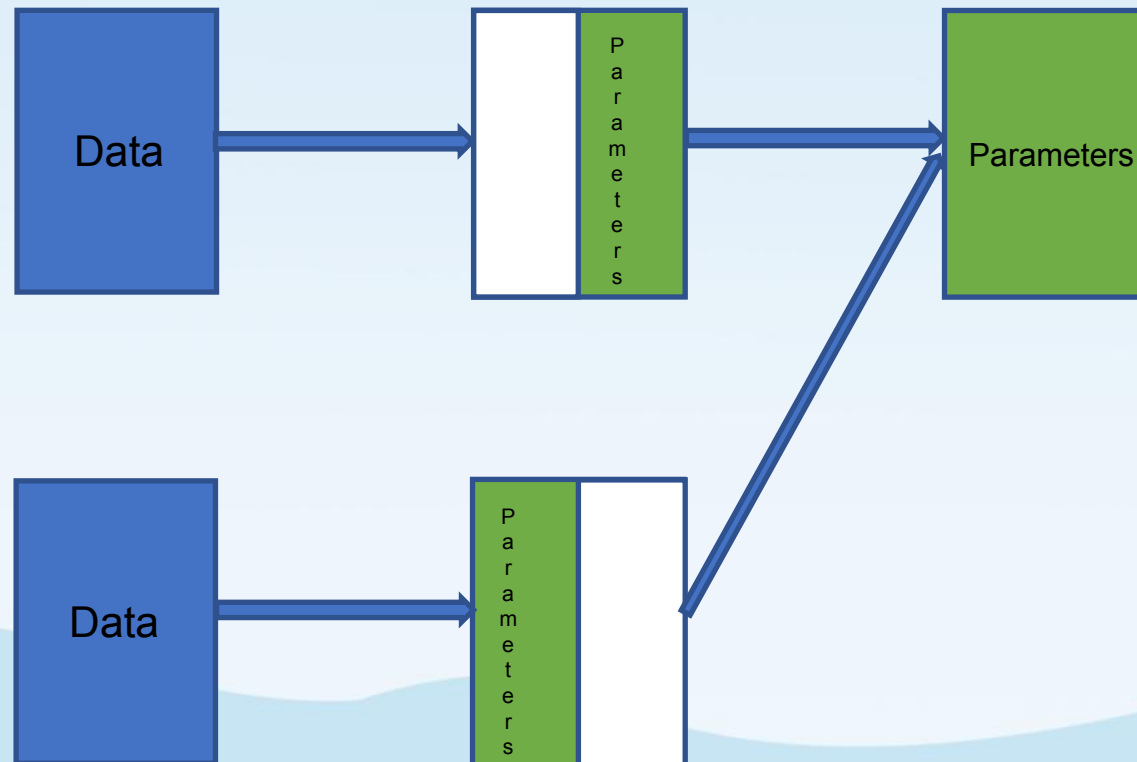
Data Parallelism

Each GPU has the same model copy, and the dataset is split into multiple copies for training on different GPUs. After each iteration of training is completed, each GPU needs to globally synchronize the gradients calculated in reverse, and then calculate the parameters used in the next iteration. In data parallelism, it is necessary to perform an Allreduce on the gradients of each GPU on the network and merge the gradients.



Model Parallelism

Model parallelism is often used to solve the problem of insufficient memory for a single node. Taking the GPT-3 model with 175 billion parameters as an example, if each parameter in the model is represented by a 16 bit floating-point number, the model would require approximately 350GB (175G x 2 bytes) of memory. The NVIDIA A100 also only supports 80GB of graphics memory and cannot fully fit the entire model into it.



Part 02

Container Network Requirements

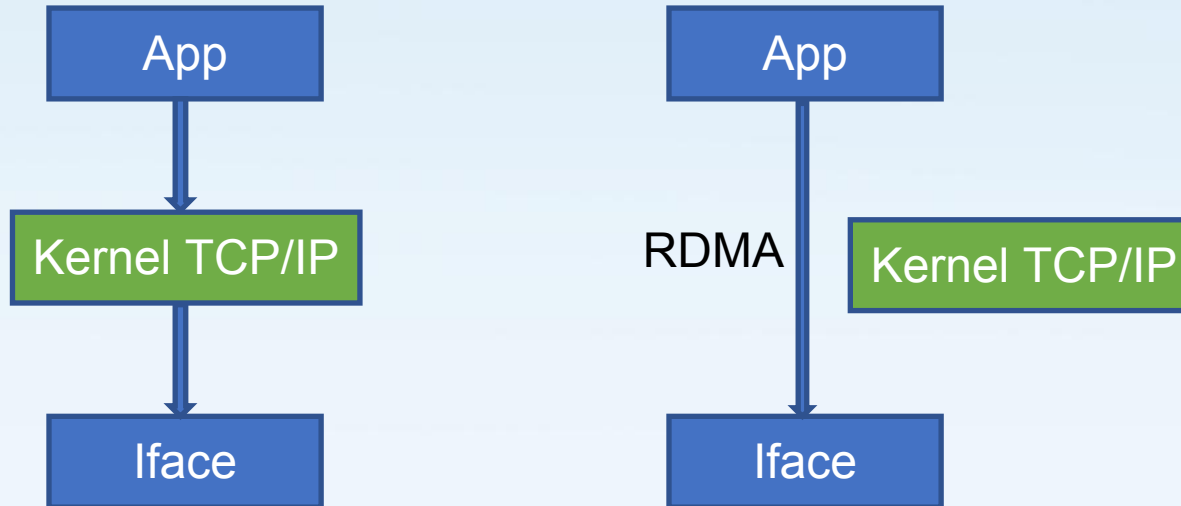


RDMA (Remote Direct Memory Access)

Infiniband: This solution has redesigned the protocol stack to use a dedicated InfiniBand, which is costly but performs the best.

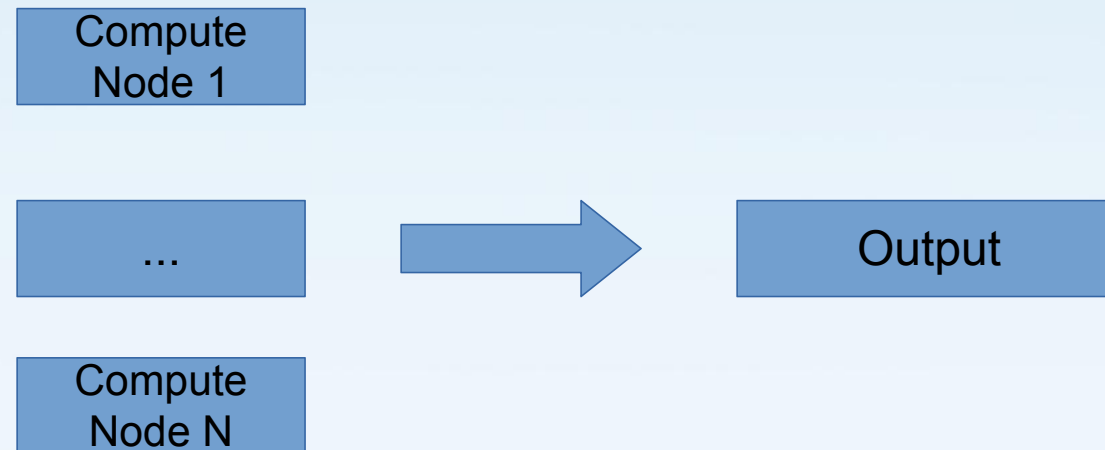
Internet Wide Area RDMA Protocol(iWARP): RDMA network based on TCP

RDMA over Converged Ethernet (RoCE): RoCEv2 is an RDMA network based on the UDP protocol.



Distributed Training

```
torchrun --nproc_per_node=4 --nnodes=2 --node_rank=0 \  
    --master_addr="192.0.0.1" --master_port=1234 ... trian.py  
torchrun --nproc_per_node=4 --nnodes=2 --node_rank=1 \  
    --master_addr="192.0.0.1" --master_port=1234 ... trian.py
```



Network backend

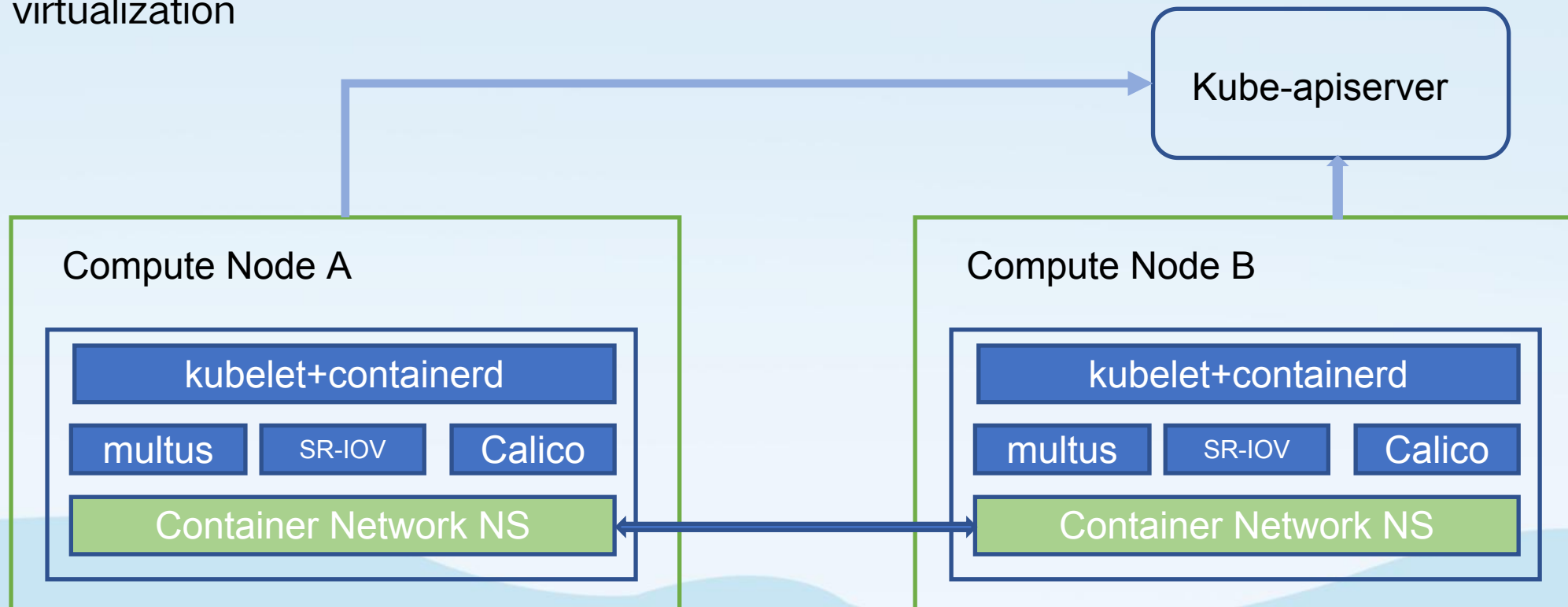
There are various types of network hardware in distributed clusters, including Ethernet, InfiniBand networks, etc. Deep learning frameworks such as PyTorch typically do not directly manipulate hardware, but instead use communication libraries. Common communication libraries include MPI, GLOO, NCCL, etc., which can be selected and configured according to specific situations.

Parameters

- **backend** (*str or Backend, optional*) – The backend to use. Depending on build-time configurations, valid values include `mpi`, `gloo`, `nccl`, and `ucc`. If the backend is not provided, then both a `gloo` and `nccl` backend will be created, see notes below for how multiple backends are managed. This field can be given as a lowercase string (e.g., `"gloo"`), which can also be accessed via `Backend` attributes (e.g., `Backend.GLOO`). If using multiple processes per machine with `nccl` backend, each process must have exclusive access to every GPU it uses, as sharing GPUs between processes can result in deadlocks. `ucc` backend is experimental.

Container network design

- Building a metadata exchange network based on Calico architecture and a training network based on RDMA network
- With the help of multiple CNIs, SRIOV(Spiderpool) supports the allocation of RDMA iface virtualization



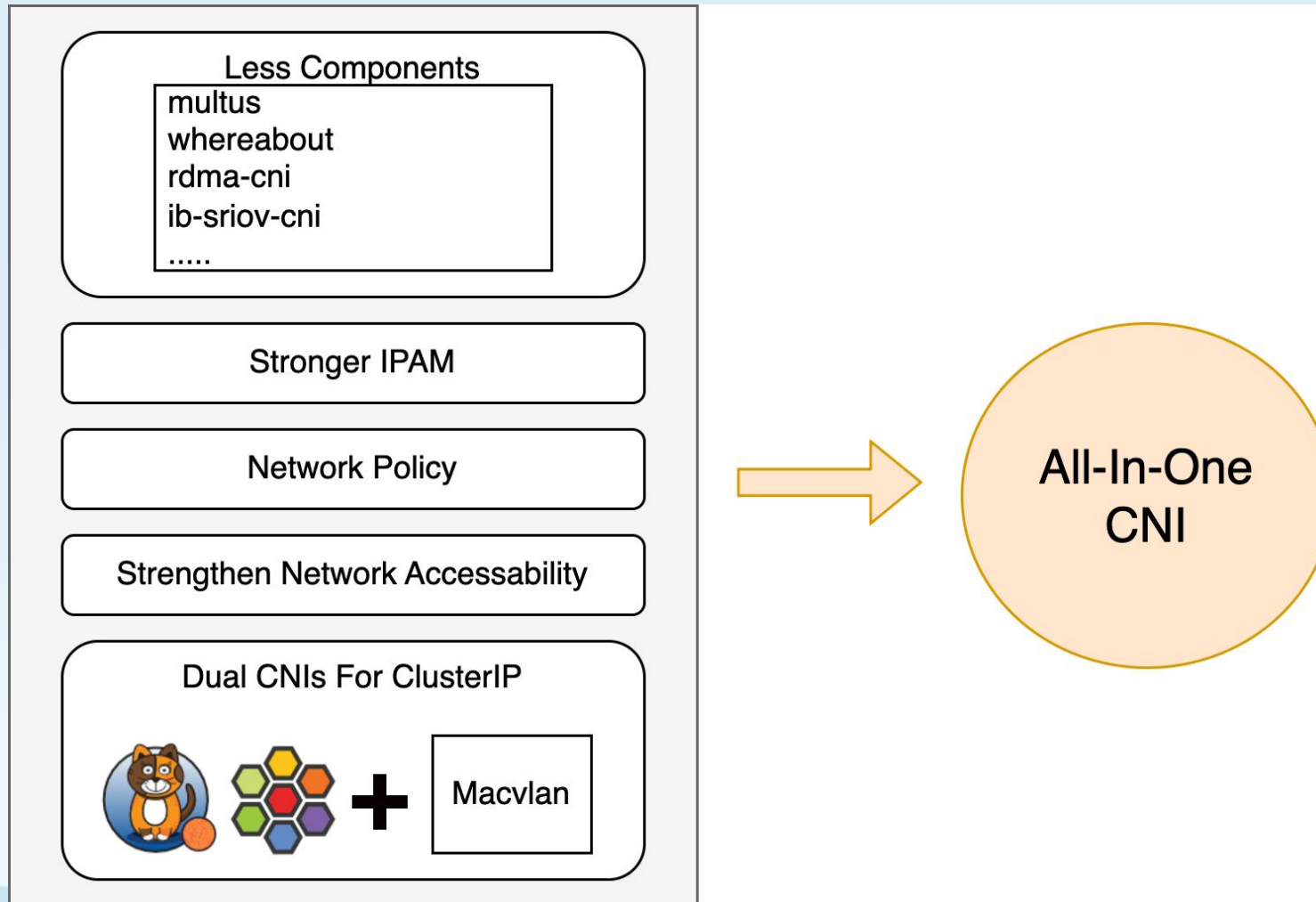


Part 03

Spliderpool

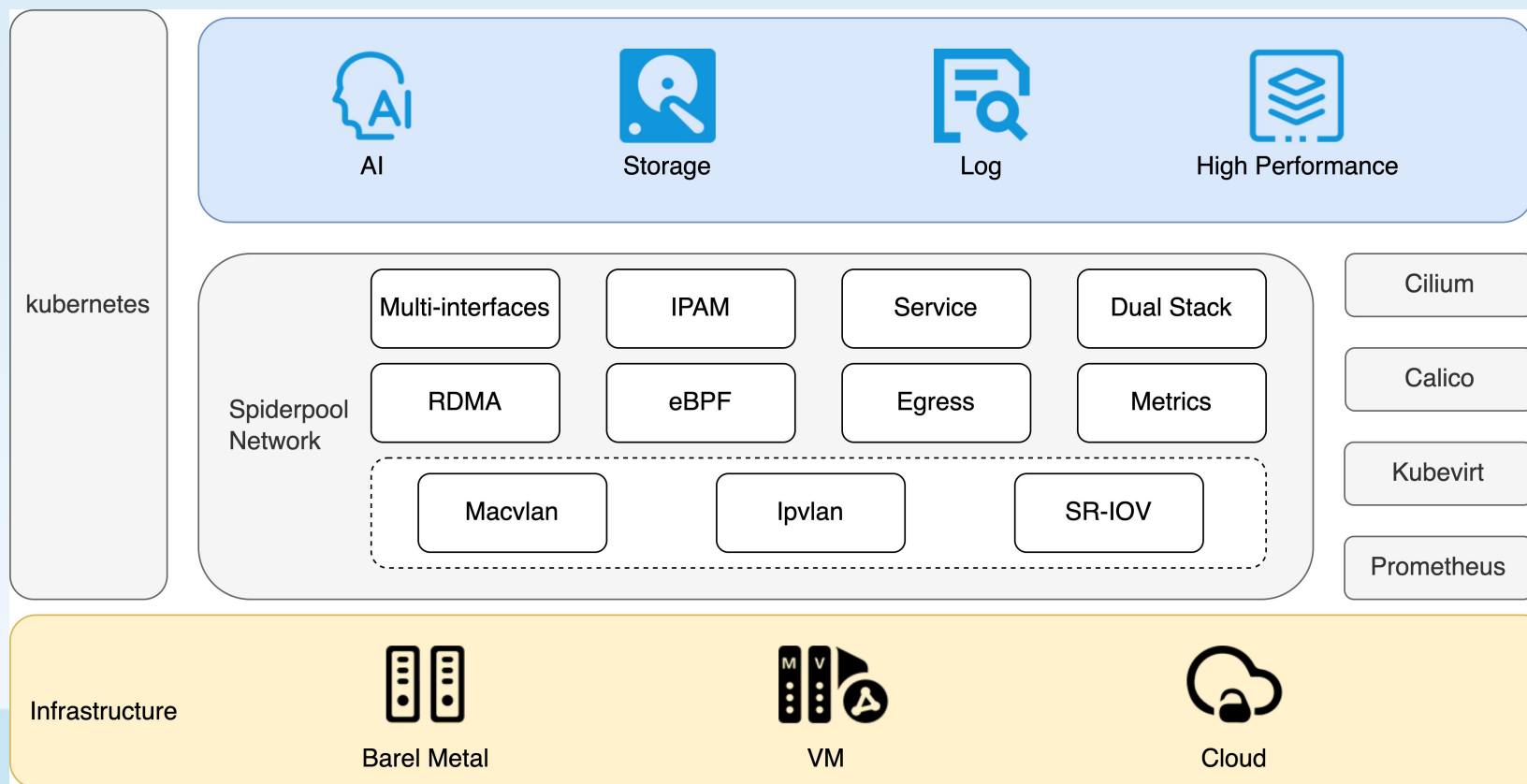


All-In-One





Spiderpool

As a **CNCF sandbox project**, Spiderpool is an **underlay and RDMA** network solution of the Kubernetes, and particularly benefits network I/O-intensive and low-latency applications like storage, middleware, and AI. It could run on any environments such as **bare metal, virtual machine, and public cloud**.



Comparison

	Spiderpool	Network Operator
SR-IOV / Macvlan / Ipvlan	✓	✓
RDMA (RoCE / Infiniband)		
IPAM		
Access Service	✓	×
Network Policy	✓ (ipvlan)	×
Bandwidth	✓ (ipvlan)	×
Submariner	✓	×
Egress Policy	✓	×
VPC	✓	×

Infiniband or RoCE

RoCEv2 is good enough, Infiniband is better

➤ Performance

InfiniBand offers superior network performance, especially in large-scale clusters.

➤ Scale

InfiniBand effectively supports clusters with tens of thousands of GPUs, while thousands of GPUs for RoCE

➤ Maintenance

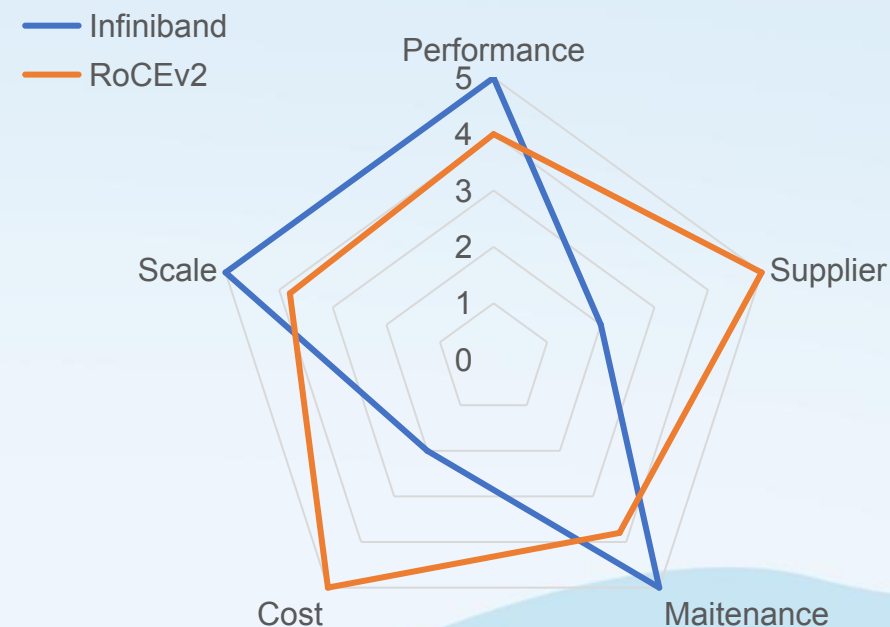
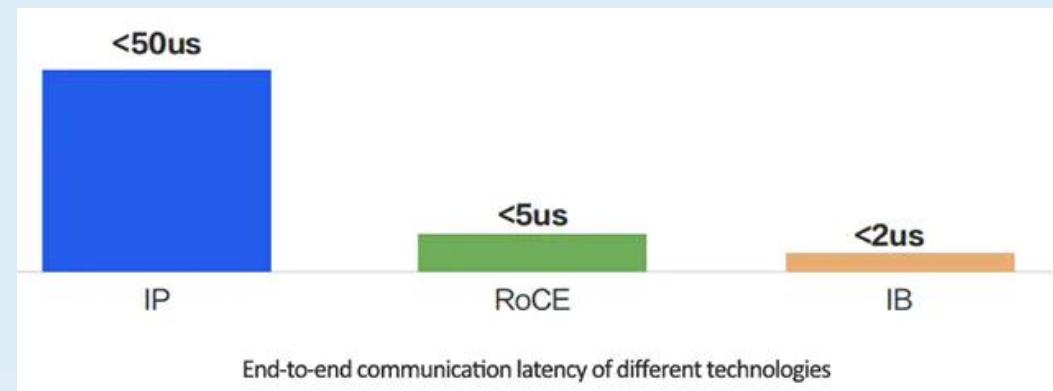
InfiniBand offers more mature management using Nvidia's UFM.

➤ Cost

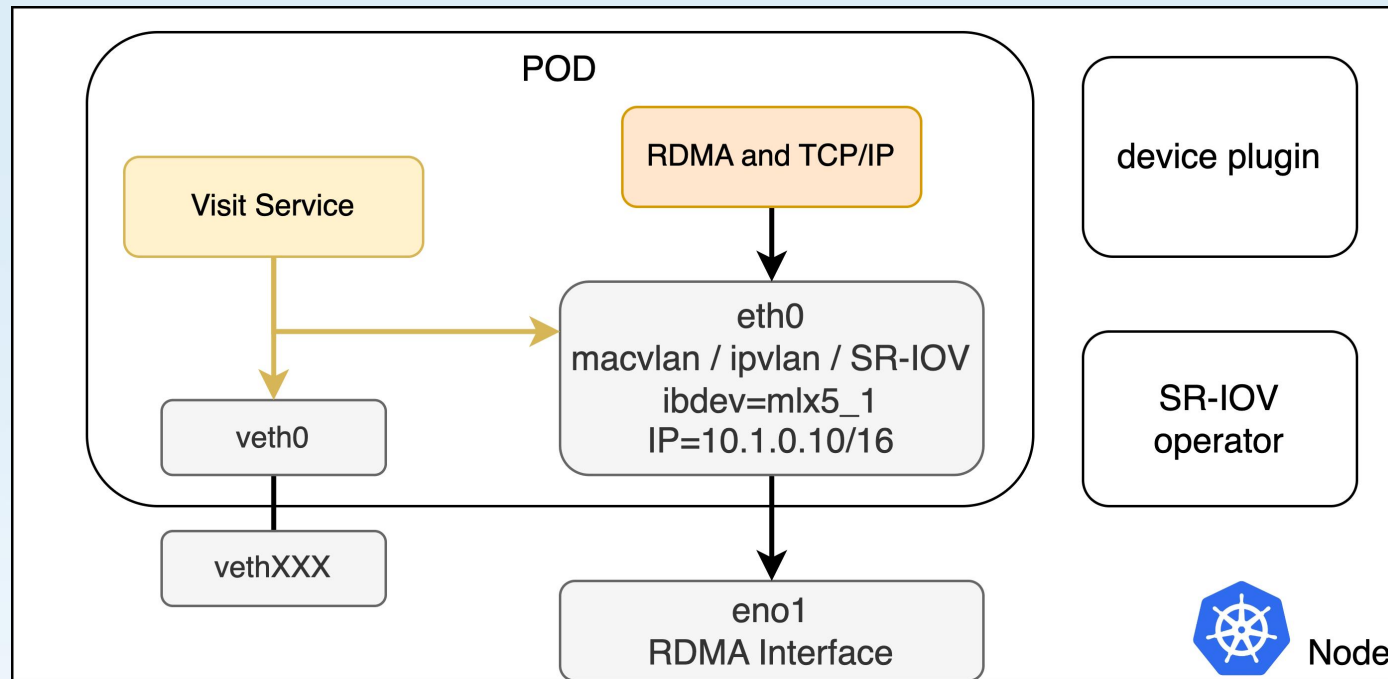
InfiniBand network hardware is more expensive.

➤ Vendor

Nvidia nearly monopolizes the InfiniBand network equipment market. RoCE has a wider range of vendor choices.



Interface



Enhance the network accessibility of macvlan, ipvlan and SR-IOV :

- Connectivity between Pod and local node
The Pod healthy check works even when the Pod and the local node join different subnets.
- Access service
 1. NAT by kube-proxy
 2. NAT by cgroup eBPFUp to 25% improvement on network delay , up to 50% improvement on network throughput
- Accessibility check based on probing ARP when launching the Pod
 1. IP conflict check
 2. Gateway reachability check

RDMA options

		Infiniband with IPoIB		
—	—	bare metal and VM	bare metal	bare metal and VM
—	√	×	√	√
—	√	×	√	√
—	—	×	exclusive or shared	—
—	√	×	√	×
—	√	×	√	×



IPAM Feature

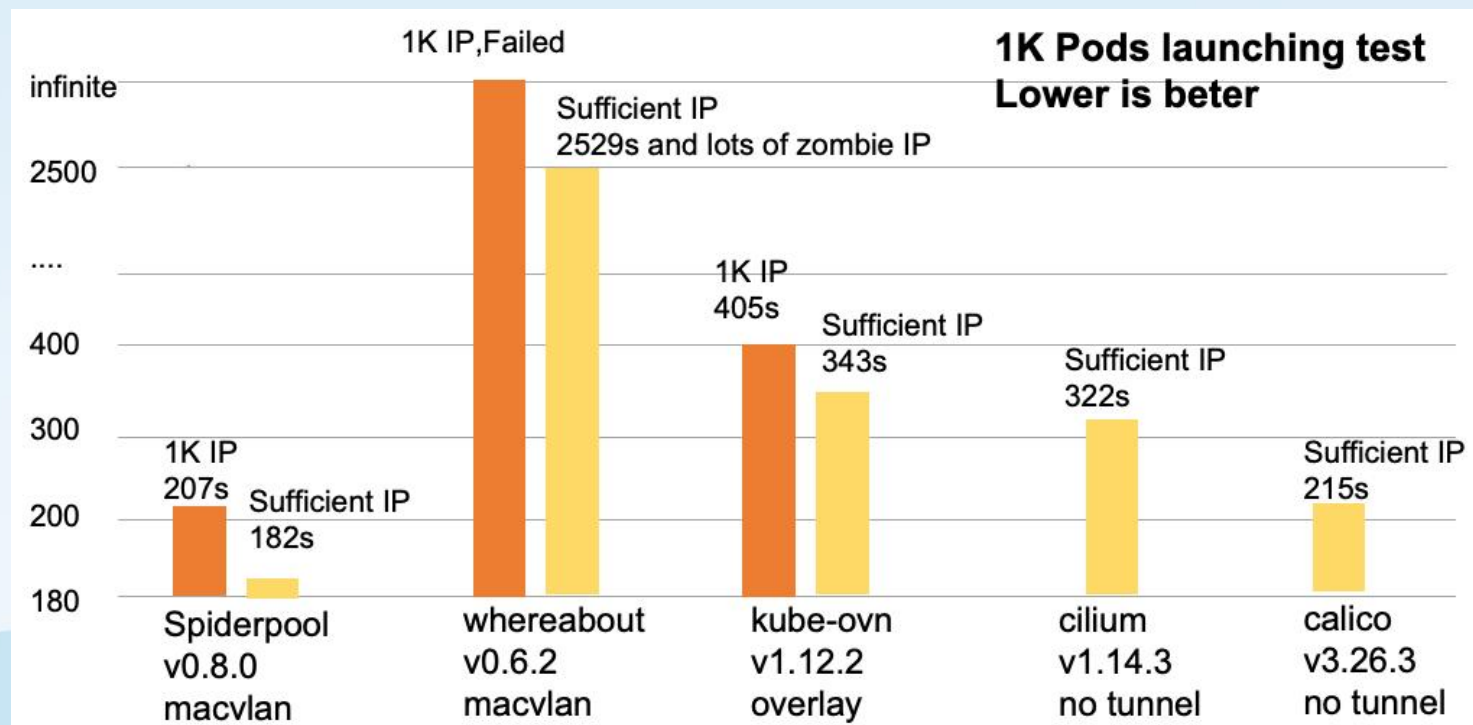
- Ippool based on CRD, IPv4-only, IPv6-only, Dual Stack
- Ippool Affinity : pod Affinity, node Affinity, namespace Affinity, multus Affinity
- Fixed IP address for stateless and stateful workload
- Support to automatically manage ippool by creating, scaling and deleting, along with application
- Reclaim leaked IP for guaranteeing IP address available for running new Pod
- Assign separate IP address for multiple underlay interfaces
- Support VPC



IPAM Performance Test

A test calculates the time cost to launch 1K Pods with the 1K IP or sufficient IP, which evaluates IPAM efficiency and stability.

- No IP conflict and wastage
- Good efficiency of assigning and relasing IP



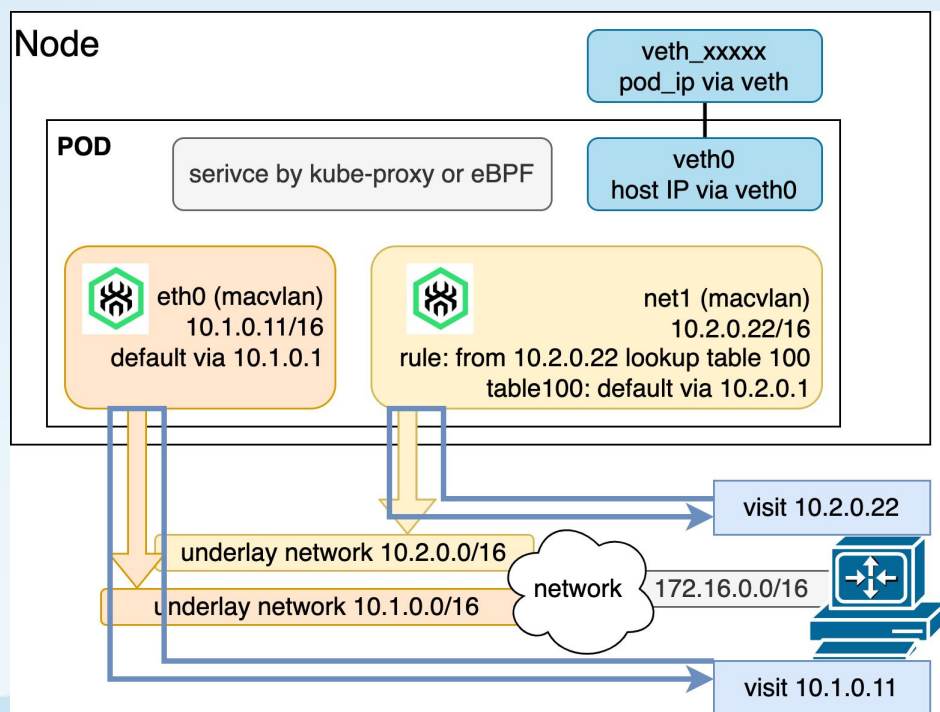
Environment :

- VM with 3 C / 8 G
- 10 VMs including 3 masters
- Kubernetes v1.26.7
- Assign IPv4 and IPv6 IP to Pod
- Launch 100 deployments with 10 replicas each

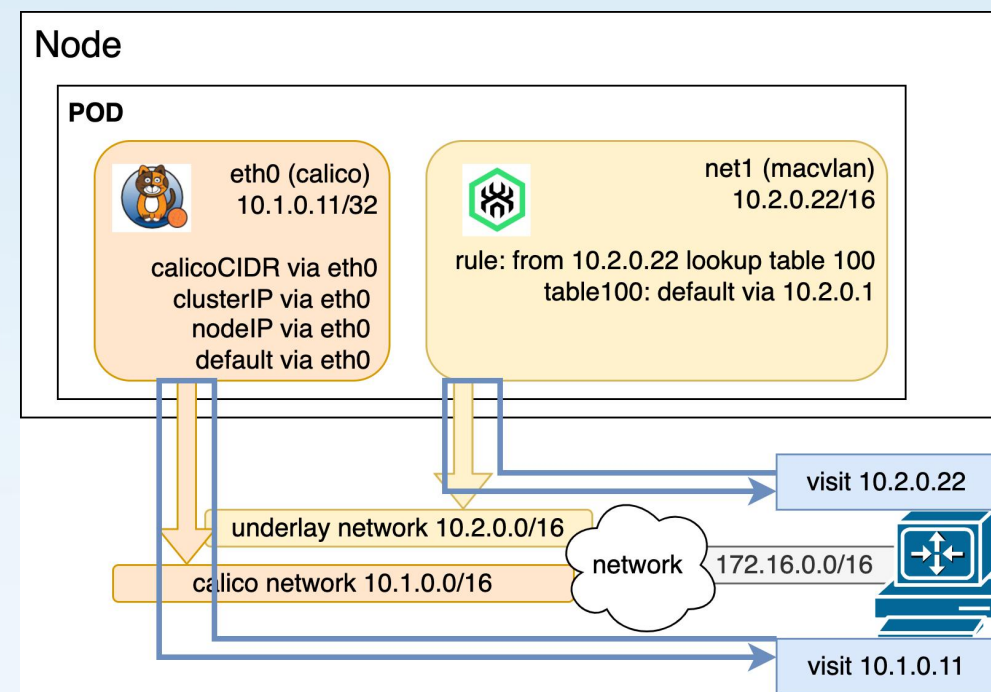
Single And Dual CNIs

Spiderpool could work alone or cooperate with overlay CNI well

- IPAM: Support to specify IP address for each underlay interface.
- Tune the routes for all interfaces: Generate the policy rule and move related gateway route to non-main route table. This guarantees the consistent data path of request and reply packets, avoiding packet loss.



multiple interfaces from spiderpool



multiple interfaces from spiderpool and calico

Roadmap

- DRA
Schedule Pod by network topology and statistics, which helps improve the network throughput and reduce training time of machine learning
- Strengthen RDMA
 - Bandwidth
 - Network policy
 - Observability





Thanks.

