KubeCon | CloudNativeCon

North America 2019

KubeCon | CloudNativeCon

North America 2019

Alcor

Hyperscale Cloud Network Management

Futurewei Technologies

Introduction

Alcor is a cloud native SDN platform that aims to provides high availability, high performance, and large-scale virtual networking control plane and management plane at a high resource provisioning rate.

P Performance	A Availability	S Scalability	E Extensibility
• Throughput-optimal design to allow batched provisioning of network resources • Fast provisioning path to support time-critical applications such as serviceless	• Always-on control plane without a single point of failure • Cross-AZ resilience for services and data • Fault-tolerant design with multiple resource provisioning paths	• Management of large numbers of network resources • Scale to half a million hosts and tens of millions network ports	• Unified network management of both VMs and containers • Plug-able model to support various implementations of data plane

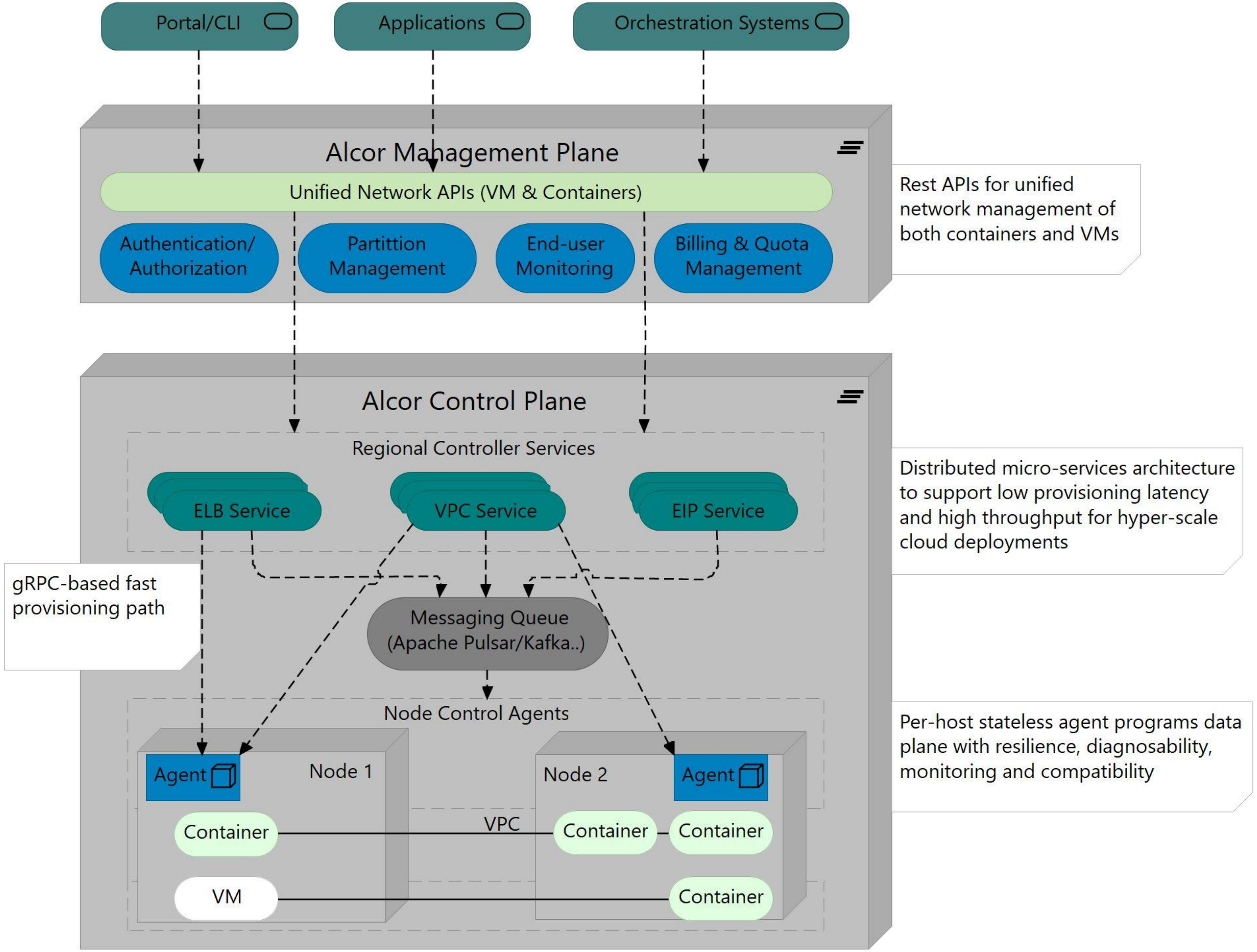
Architecture Overview

Alcor Management plane

- Expose REST APIs to clients
- Provide partition management, end-user monitoring, and billing & quota management

Alcor Control Plane

- Offer multiple network management services including VPC, ELB and EIP
- Support multiple provisioning path including fast path, normal path and rescue path
- Drive network configurations to on-host Alcor agents in scale
- Alcor agents program data plane to establish network connection between containers and VMs in the same VPC



Rest APIs for unified network management of both containers and VMs

Distributed micro-services architecture to support low provisioning latency and high throughput for hyper-scale cloud deployments

gRPC-based fast provisioning path

Per-host stateless agent programs data plane with resilience, diagnosability, monitoring and compatibility

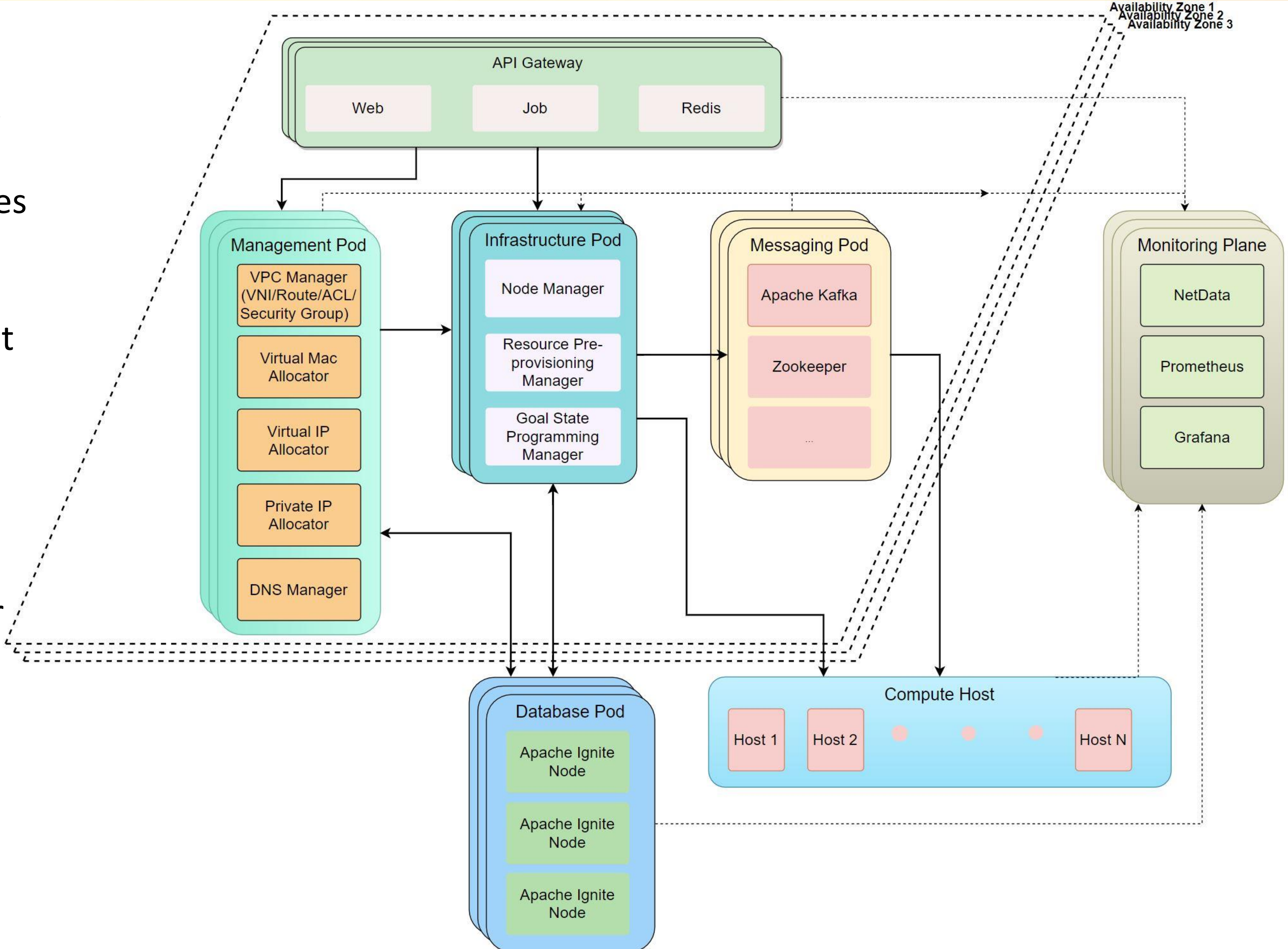
Cloud-Native Control Plane

Powered by Kubernetes

- Each controller instance is an Kubernetes application
- Each application contains multiple services
  - Customer resource management service
  - Infrastructure resource management service
  - Database service
  - Messaging service

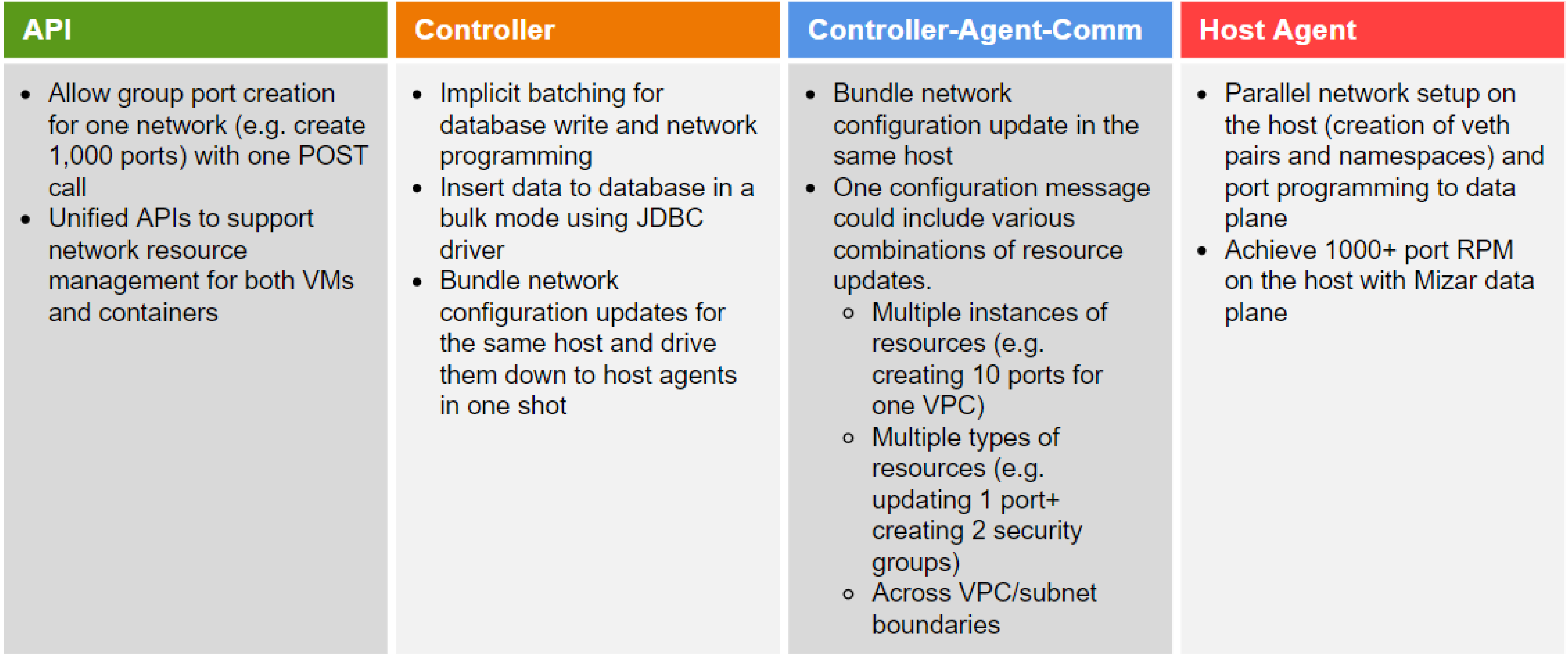
Distributed Micro-Services Architecture

- Use Istio to secure, connect, and monitor control plane micro-services
- Fine-grained control of service-to-service communication including load balancing, retries, failovers, and rate limits.



Throughput-Optimal Design

Focus on throughout optimization on every system layer



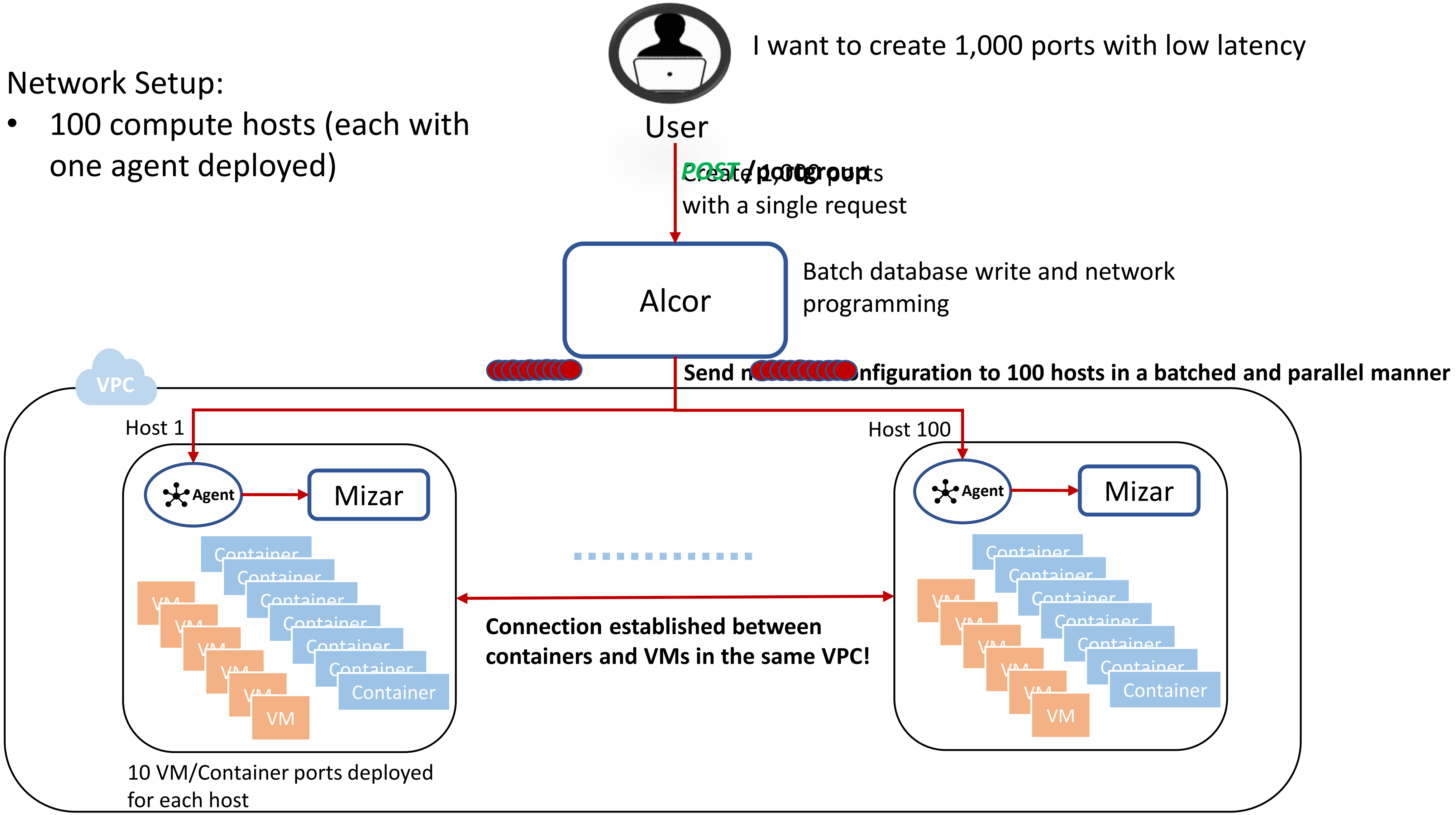
API	Controller	Controller-Agent-Comm	Host Agent
• Allow group port creation for one network (e.g. create 1,000 ports) with one POST call • Unified APIs to support network resource management for both VMs and containers	• Implicit batching for database write and network programming • Insert data to database in a bulk mode using JDBC driver • Bundle network configuration updates for the same host and drive them down to host agents in one shot	• Bundle network configuration update in the same host • One configuration message could include various combinations of resource updates. ◦ Multiple instances of resources (e.g. creating 10 ports for one VPC) ◦ Multiple types of resources (e.g. updating 1 port+ creating 2 security groups) ◦ Across VPC/subnet boundaries	• Parallel network setup on the host (creation of veth pairs and namespaces) and port programming to data plane • Achieve 1000+ port RPM on the host with Mizar data plane

User Scenario: Large-Scale VPC Provisioning

KubeCon | CloudNativeCon North America 2019

Network Setup:

- 100 compute hosts (each with one agent deployed)



I want to create 1,000 ports with low latency

User

POST /portgroup

Create 1,000 ports with a single request

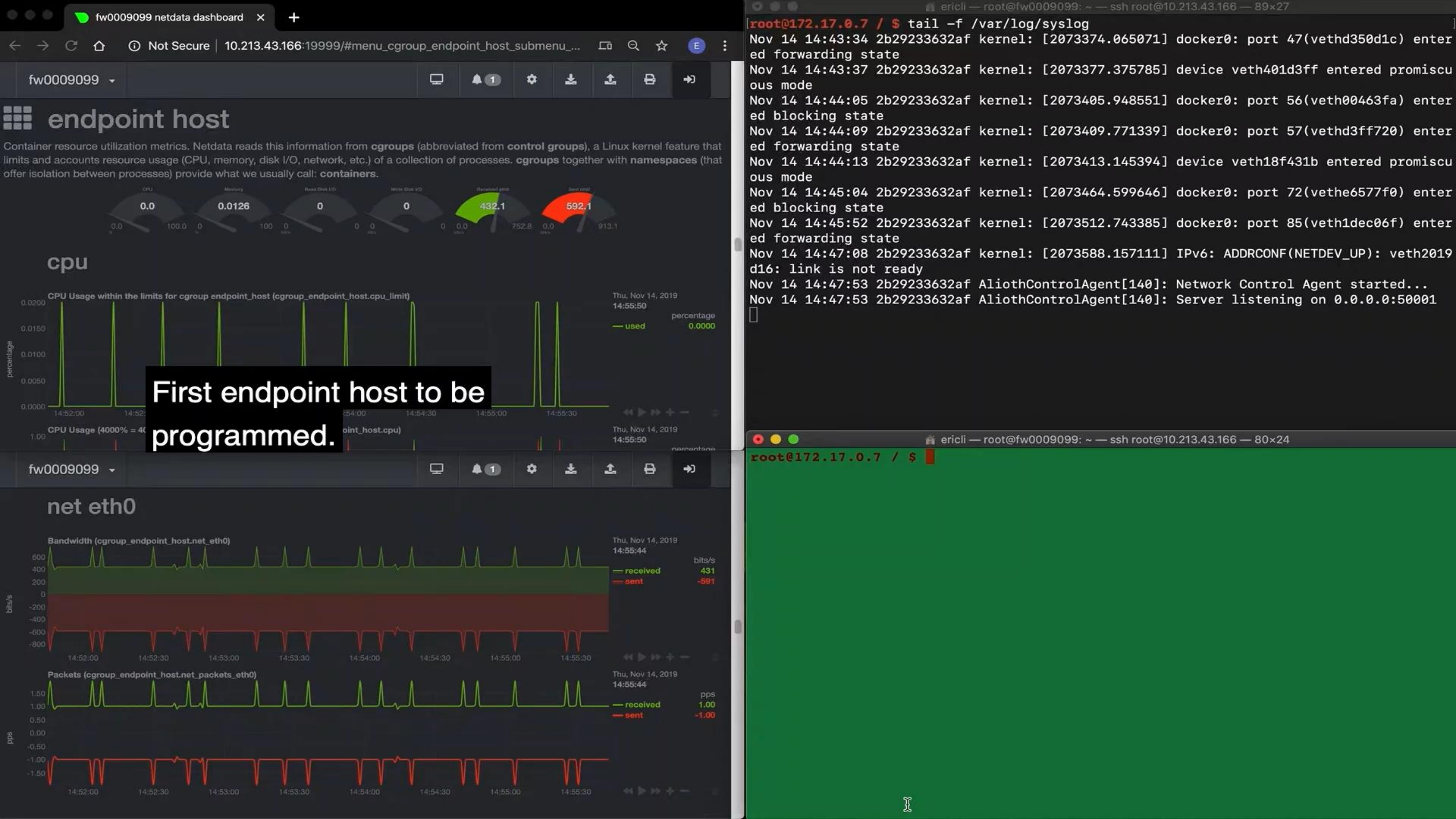
Alcor

Batch database write and network programming

Send n nfiguration to 100 hosts in a batched and parallel manner

Connection established between containers and VMs in the same VPC!

10 VM/Container ports deployed for each host



fw0009099 netdata dashboard
Not Secure | 10.213.43.166:19999/#menu_cgroup_endpoint_host_submenu_...
fw0009099
CPU
Memory
Read Disk I/O
Write Disk I/O
Received eth0
Sent eth0
0.0
0.0126
0
0
432.1
592.1
cpu
CPU Usage within the limits for cgroup endpoint_host (cgroup_endpoint_host.cpu_limit)
Thu, Nov 14, 2019
14:55:50
percentage
used 0.0000
CPU Usage (4000% = 40 ... oint_host.cpu)
net eth0
Bandwidth (cgroup_endpoint_host.net_eth0)
Thu, Nov 14, 2019
14:55:44
bits/s
received 431
sent -591
Packets (cgroup_endpoint_host.net_packets_eth0)
pps
received 1.00
sent -1.00
14:52:00
14:52:30
14:53:00
14:53:30
14:54:00
14:54:30
14:55:00
14:55:30
ericli — root@fw0009099: ~ — ssh root@10.213.43.166 — 89×27
ericli — root@fw0009099: ~ — ssh root@10.213.43.166 — 80×24

endpoint host

Container resource utilization metrics. Netdata reads this information from **cgroups** (abbreviated from **control groups**), a Linux kernel feature that limits and accounts resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes. **cgroups** together with **namespaces** (that offer isolation between processes) provide what we usually call: **containers**.

```
root@172.17.0.7 / $ tail -f /var/log/syslog
Nov 14 14:43:34 2b29233632af kernel: [2073374.065071] docker0: port 47(vethd350d1c) entered forwarding state
Nov 14 14:43:37 2b29233632af kernel: [2073377.375785] device veth401d3ff entered promiscuous mode
Nov 14 14:44:05 2b29233632af kernel: [2073405.948551] docker0: port 56(veth00463fa) entered blocking state
Nov 14 14:44:09 2b29233632af kernel: [2073409.771339] docker0: port 57(vethd3ff720) entered forwarding state
Nov 14 14:44:13 2b29233632af kernel: [2073413.145394] device veth18f431b entered promiscuous mode
Nov 14 14:45:04 2b29233632af kernel: [2073464.599646] docker0: port 72(vethe6577f0) entered blocking state
Nov 14 14:45:52 2b29233632af kernel: [2073512.743385] docker0: port 85(veth1dec06f) entered forwarding state
Nov 14 14:47:08 2b29233632af kernel: [2073588.157111] IPv6: ADDRCONF(NETDEV_UP): veth2019d16: link is not ready
Nov 14 14:47:53 2b29233632af AliothControlAgent[140]: Network Control Agent started...
Nov 14 14:47:53 2b29233632af AliothControlAgent[140]: Server listening on 0.0.0.0:50001
```

```
root@172.17.0.7 / $
```

First endpoint host to be programmed.

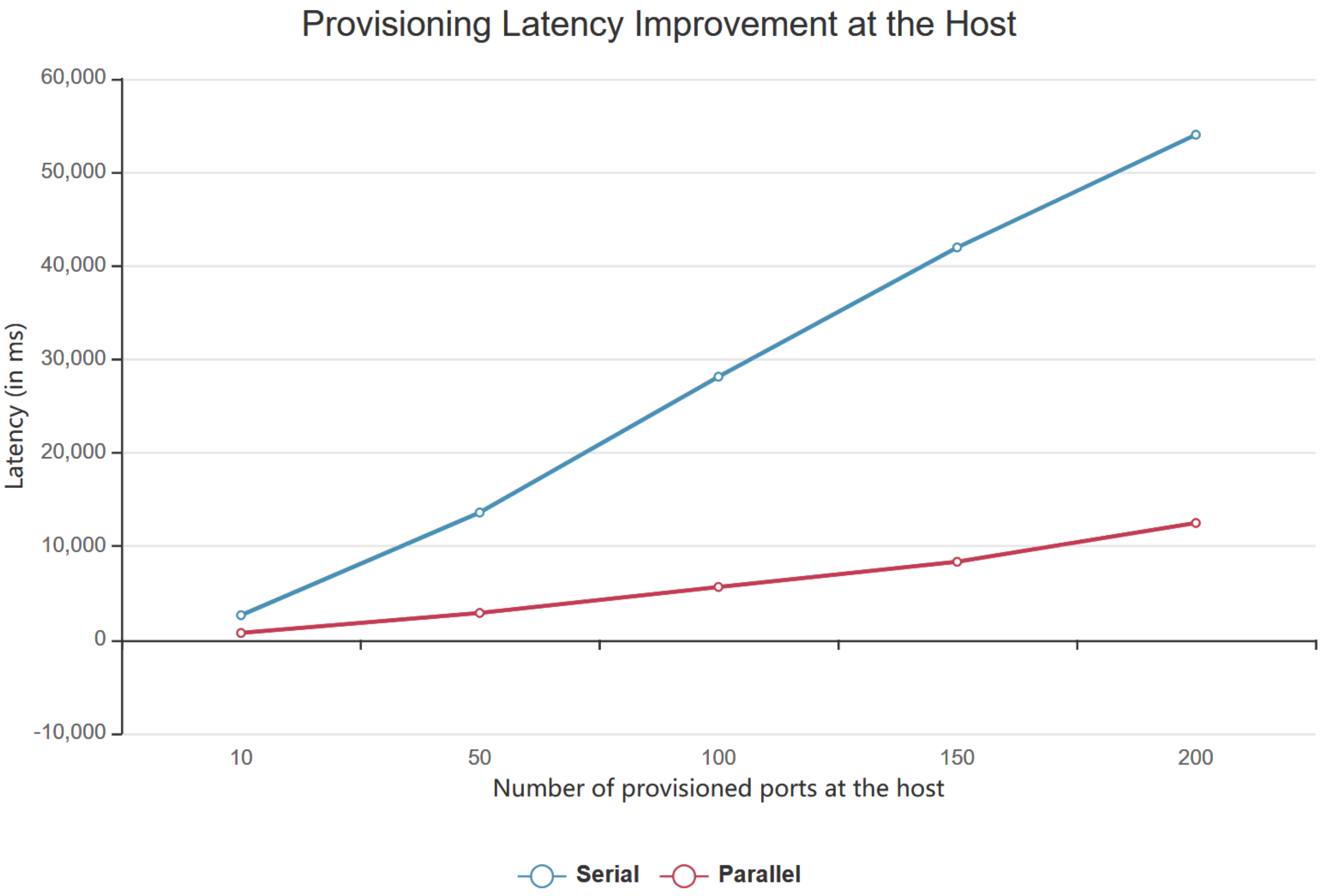
Agent Programming Latency

Serial Host Programming

- Create ports one by one including network configuration and data plane programming
- Latency increases significantly for a large number of ports

Parallel Host Programming

- Create multiple threads for network configuration (veth pair and namespace creation etc.)
- Support programming of data plane in a single-threaded mode or multi-threaded mode



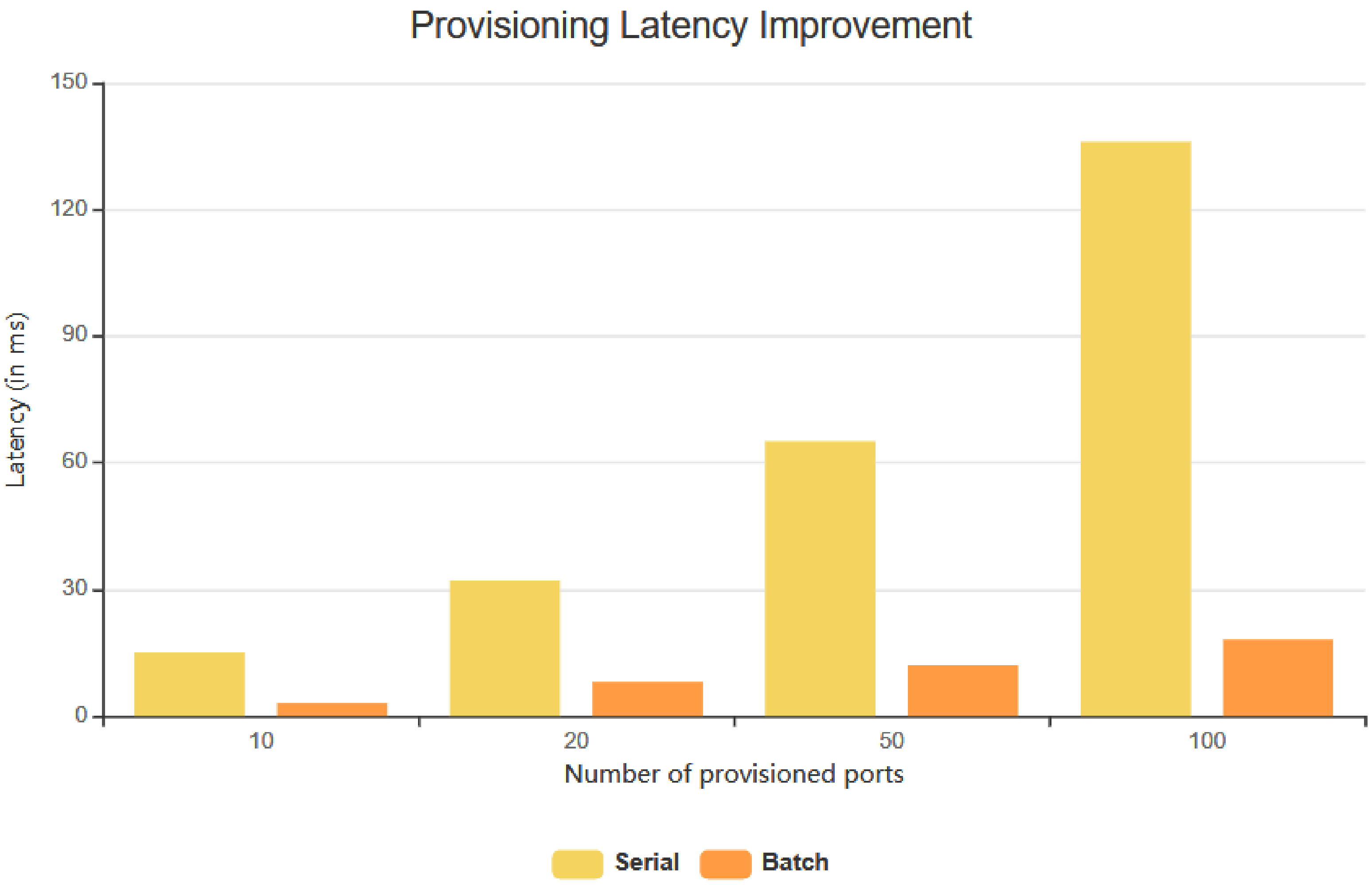
E2E Provisioning Latency

Serial Provisioning

- Ports are created and distributed one by one
- Unable to scale to a large number of ports

Batch Provisioning

- Created with a single API call (post portgroup)
- Distributed to hosts in a batched and parallel manner



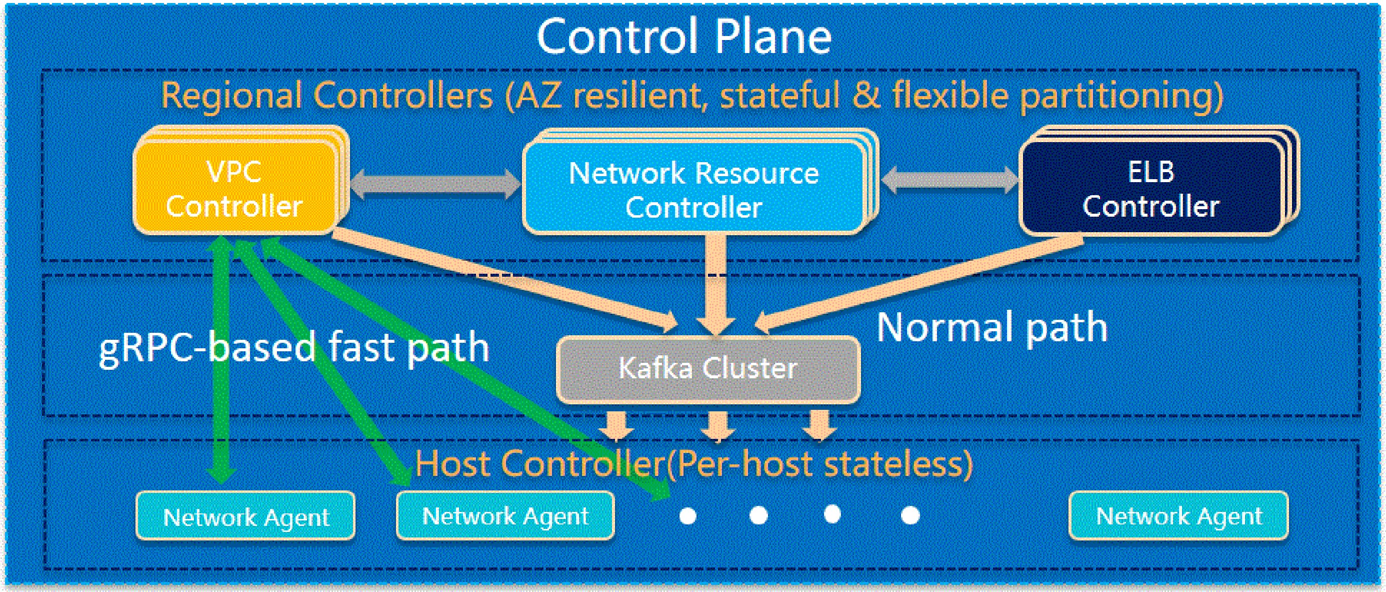
Fast Port Provisioning

Fast path design

- Direct communication channel from controller to agent
- Alternative provisioning path for control plane reliability

Background

- Customer scenarios requires ultra-low latency for E2E network configuration provisioning (in a few 10 *ms* or 100 *ms*)
- Message queue subsystem, usually adopted as a high-throughput and scalable solution for network configuration updates, may not fit into time-critical customer scenarios


Control Plane
Regional Controllers (AZ resilient, stateful & flexible partitioning)
VPC Controller
Network Resource Controller
ELB Controller
gRPC-based fast path
Kafka Cluster
Normal path
Host Controller(Per-host stateless)
Network Agent
Network Agent
Network Agent

Video

To be uploaded…