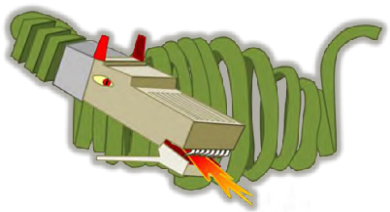
OPENSTACK DAYS
CHINA

Topic: Database Consistency Solution for large-scale OpenStack SDN Architecture

Speaker: 马力, 海云捷迅（AWcloud）
Omer Anson, 华为（Huawei）

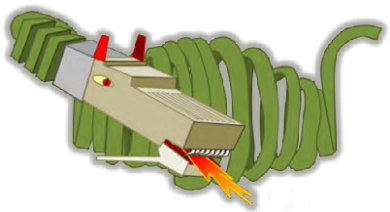
Agenda



- Introduction
- Dragonflow Overview
- What's the Problem
- How We Solve It
- To the Next Stage

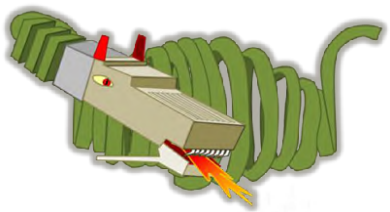
OPENSTACK DAYS CHINA

Introduction



- Li Ma
  - Principle Architect in AWcloud
  - Core in OpenStack Dragonflow
  - Concentrated on large-scale cloud infrastructure
- Omer Anson
  - Software Engineer in Huawei
  - Core in OpenStack Dragonflow

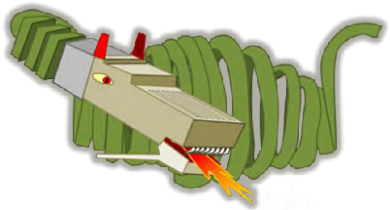
Dragonflow Overview

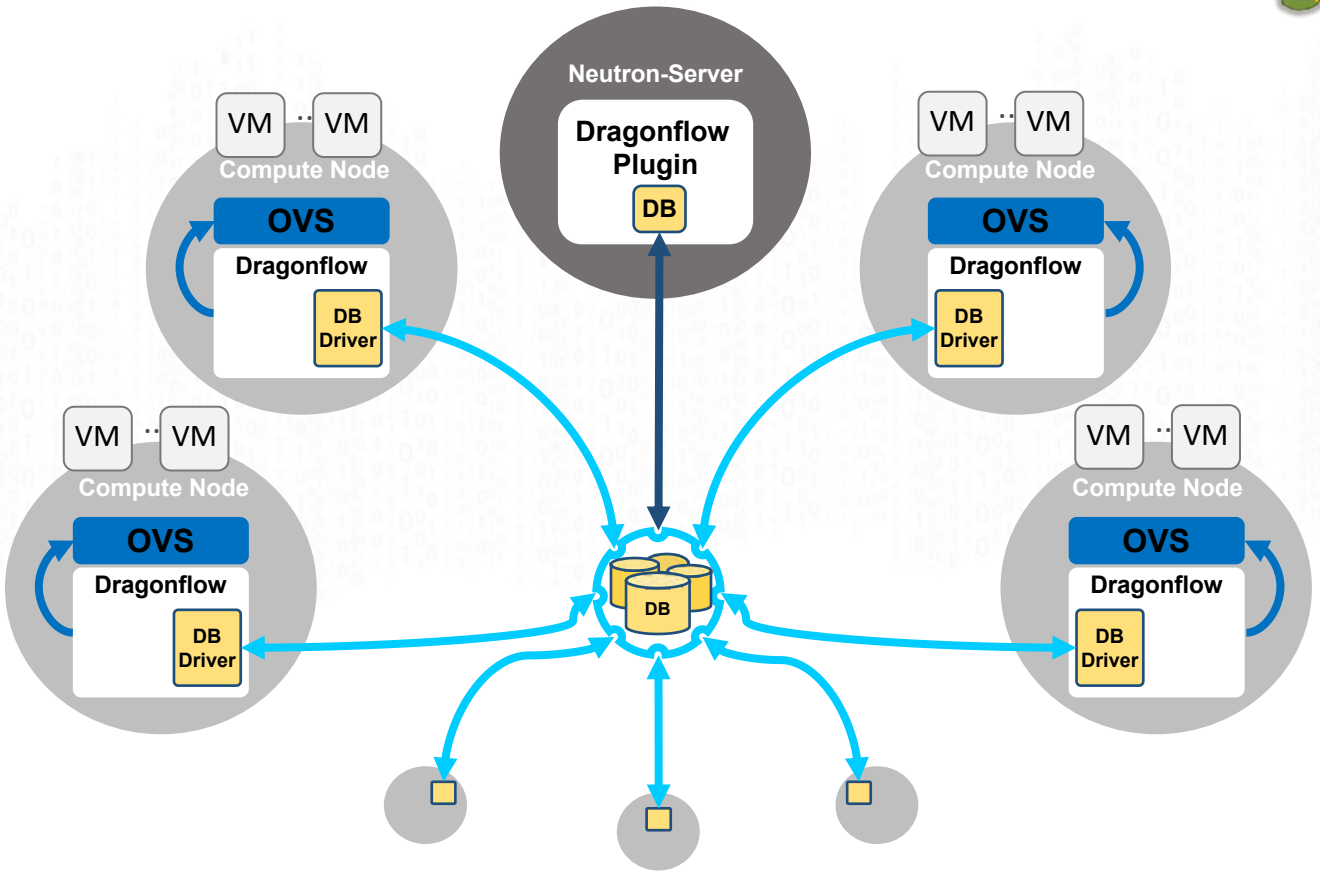


- Integral “Big Tent” project in OpenStack
- Designed for High Scale, Performance and Low Latency
- Lightweight and Simple
- Easily Extendable
- Distributed SDN Control Plane
- Focus on advanced networking services
- Distributes Policy Level Abstraction to the Compute Nodes

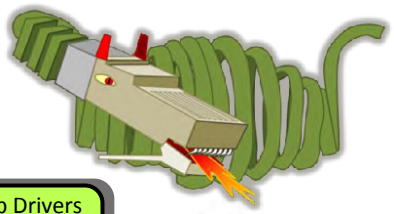
OPENSTACK DAYS CHINA

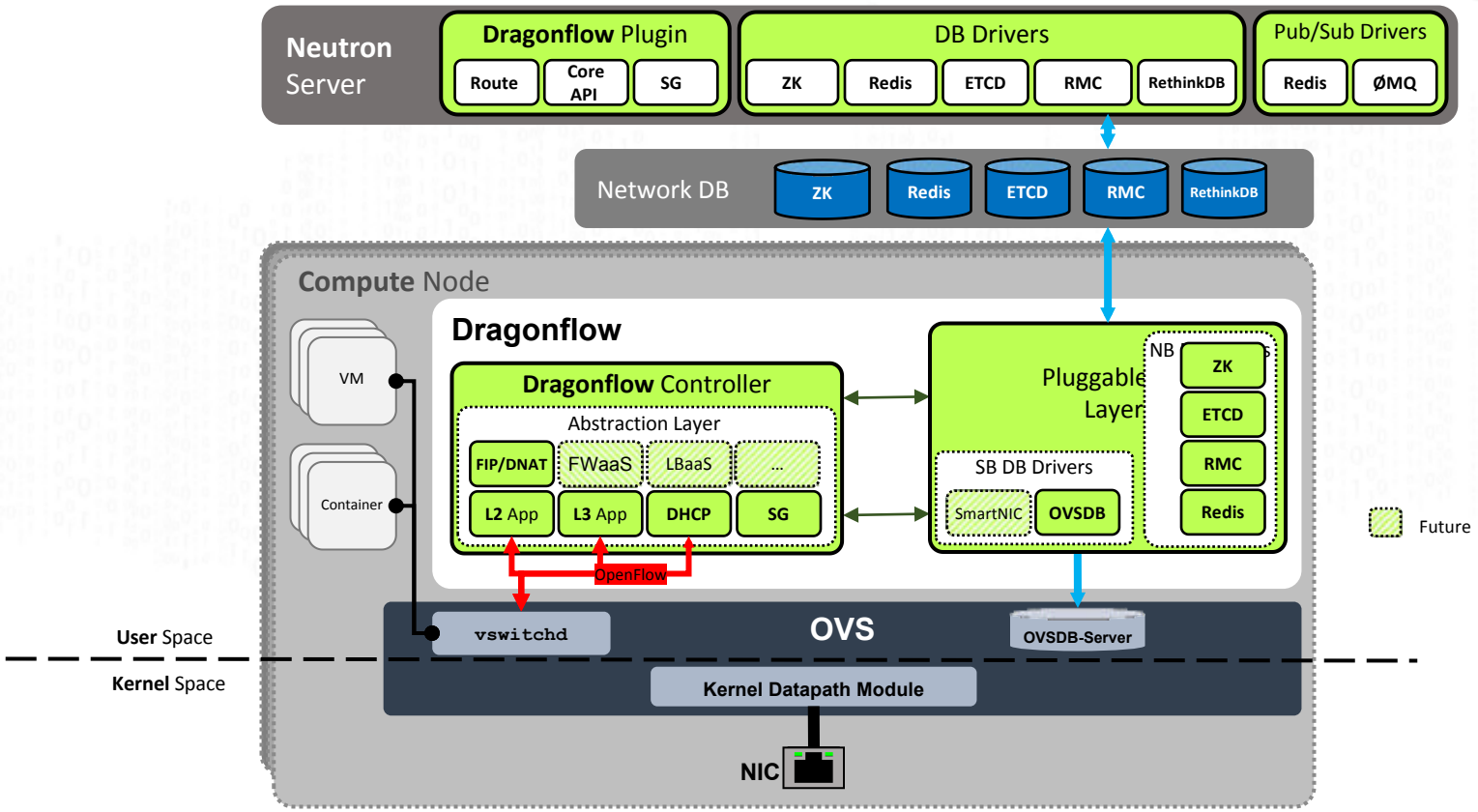
Distributed SDN





“Under The Hood”

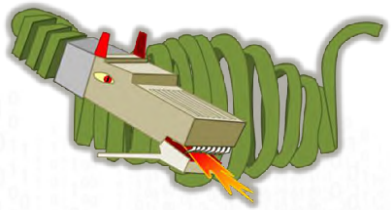




Current Release Features (**Mitaka**)

L2 core API, IPv4, IPv6

- GRE/VxLAN/STT/Geneve tunneling protocols



Distributed L3 Virtual Router
Distributed DHCP
Pluggable Distributed Database

- ETCD, RethinkDB, RAMCloud, Redis, ZooKeeper

Pluggable Publish-Subscribe

- ØMQ, Redis

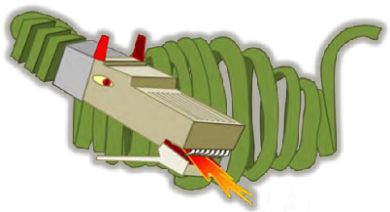
Security Groups

- OVS Flows leveraging connection tracking integration

Distributed DNAT
Selective Proactive Distribution

- Tenant Based

Pluggable Database



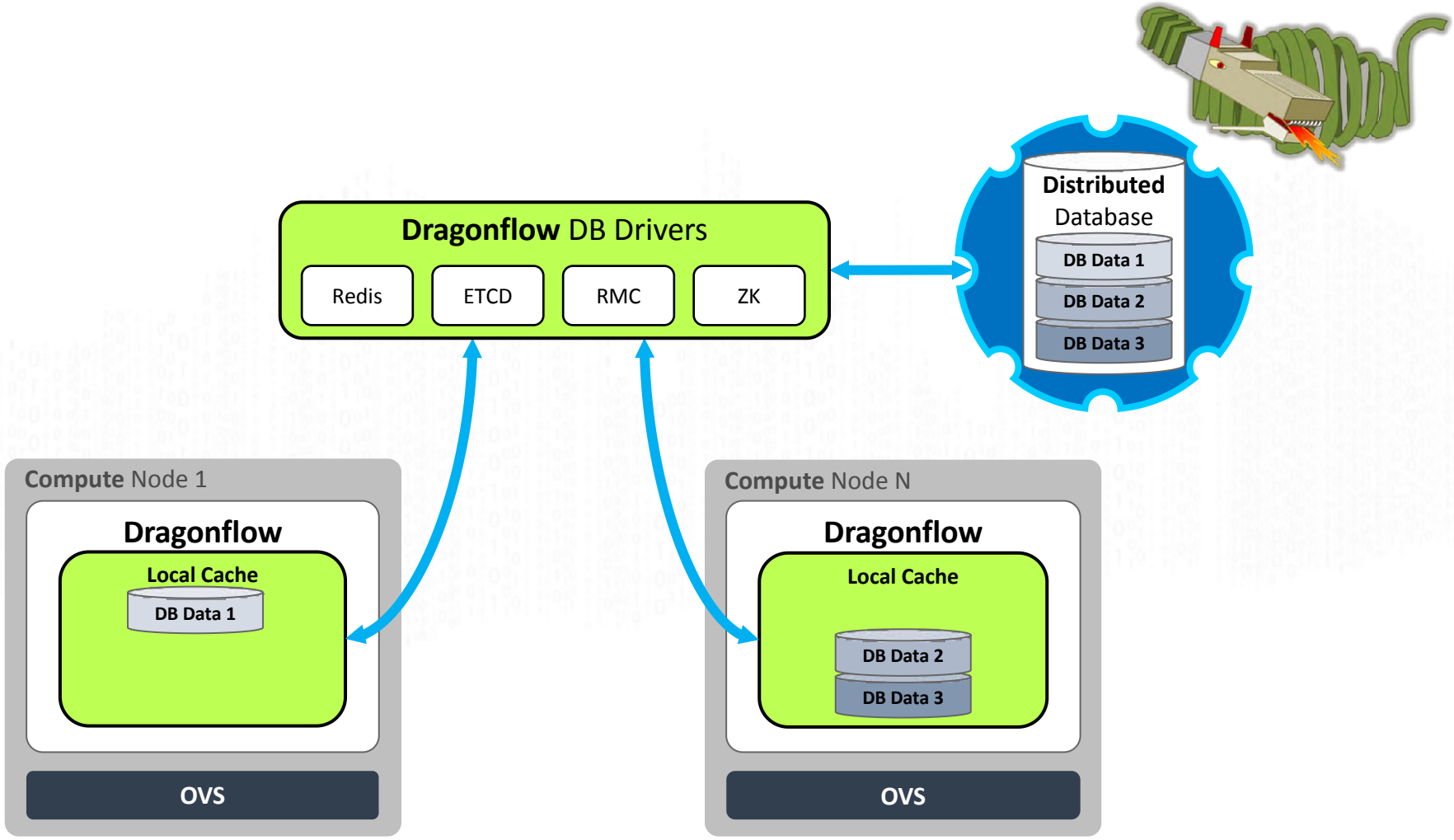
Requirements

- HA + Scalability
- Different Environments have different requirements
  - Performance, Latency, Scalability, etc.

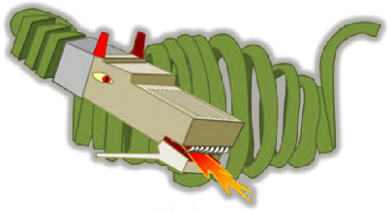
Why **Pluggable**?

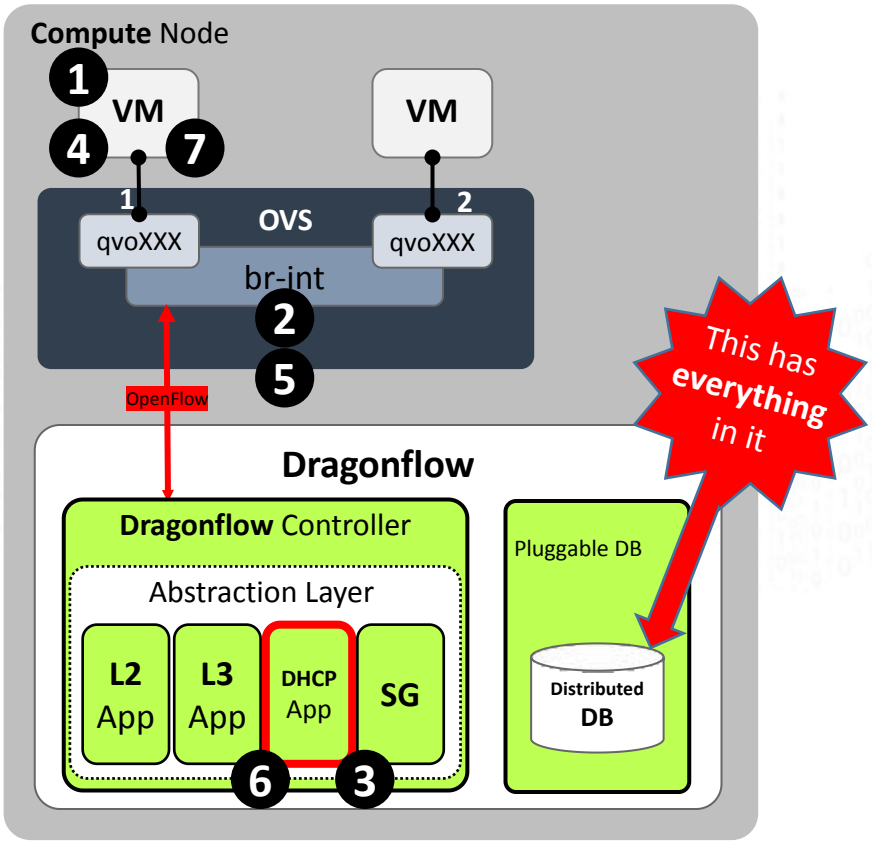
- Long time to productize
- Mature Open Source alternatives
- Allow us to focus on the networking services only

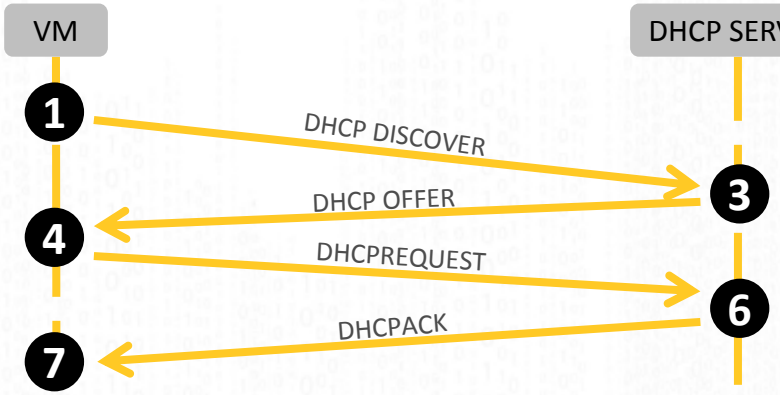
Selective **Proactive** Distribution



Distributed DHCP

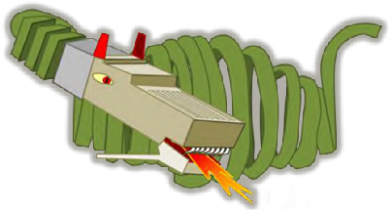


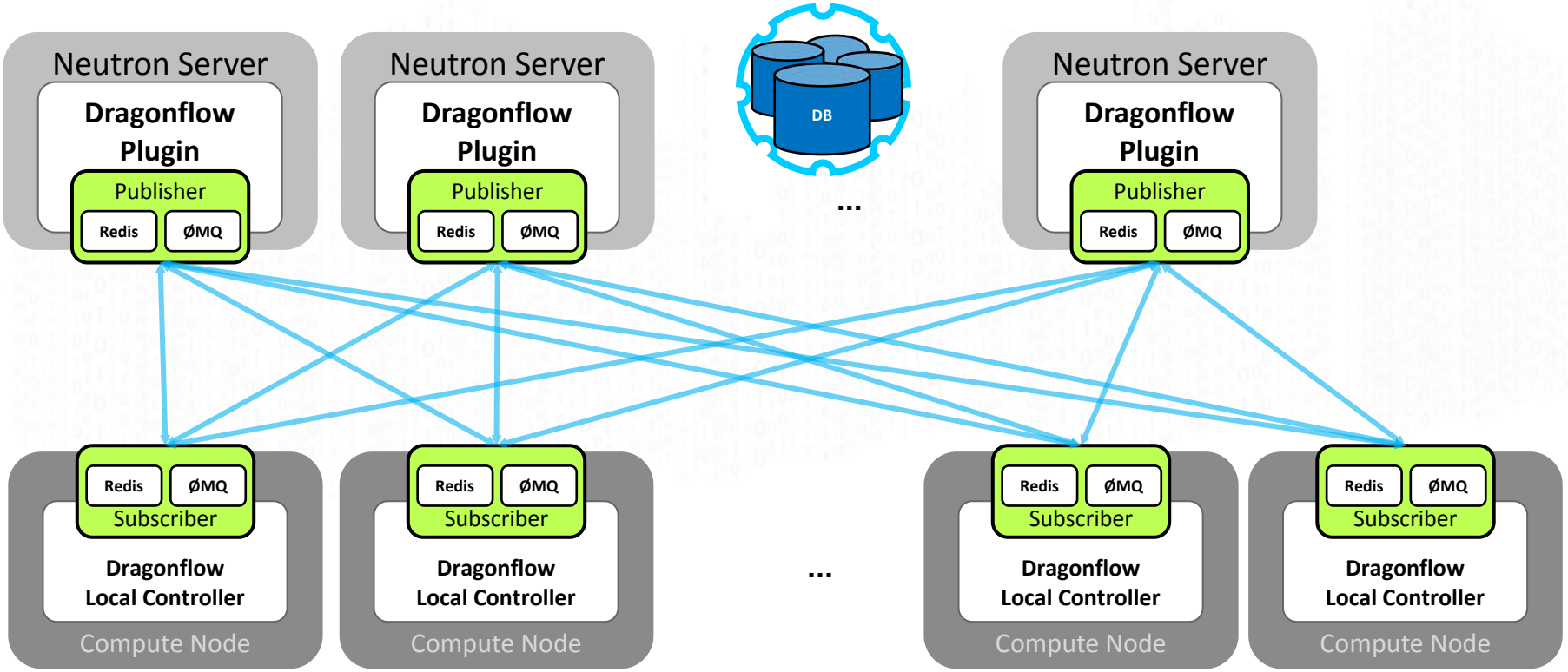




1	VM Send DHCP_DISCOVER
2	Classify Flow as DHCP, Forward to Controller
3	DHCP App sends DHCP_OFFER back to VM
4	VM Send DHCP_REQUEST
5	Classify Flow as DHCP, Forward to Controller
6	DHCP App populates DHCP_OPTIONS from DB/CFG and send DHCP_ACK

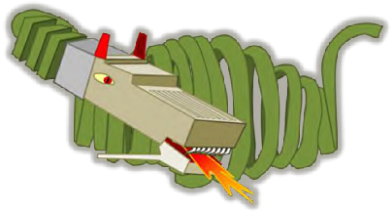
Pluggable Pub/Sub





Is Dragonflow Ready?

AWcloud Point of View



Dispatch Network Policy to Compute Nodes

Requirements:

Scalability

Reliability

Currently, we use Neutron OVS plugin

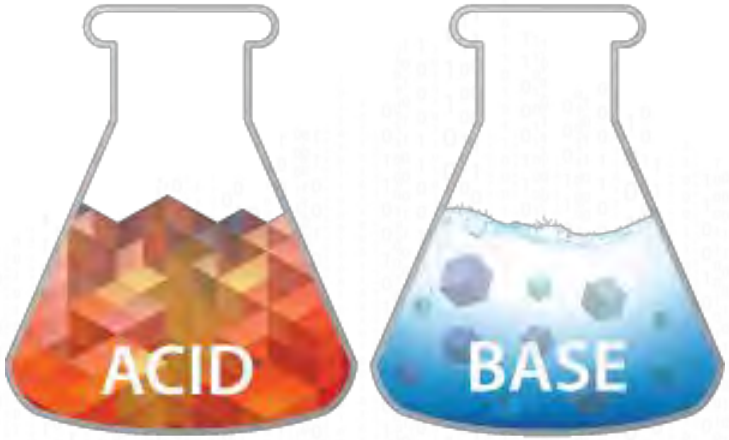
…but as workloads increase…

Limitations in Large-scale deployments

- **Messaging**
  - Distributed Messaging System for OpenStack at Scale
  - Presented in Vancouver Summit 2015
- **Persistent HA DB**
  - Dragonflow DONE the SDN way
  - Presented in Austin Summit 2016

OPENSTACK DAYS CHINA

Scalability in Persistent Storage



We prefer **BASE** systems for data backends

- Basically Available
- Soft-state
- Eventual consistent

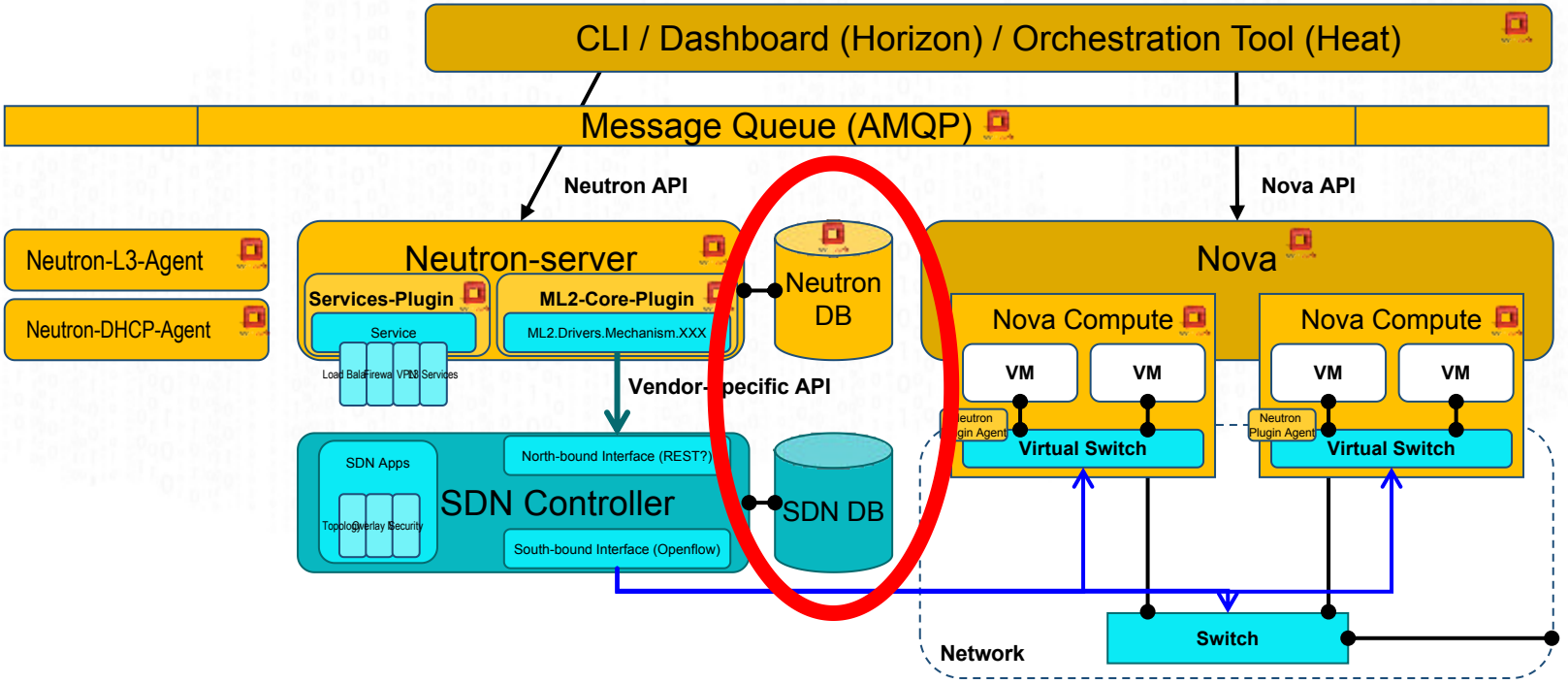
Is there any open source solution that can meet our requirements?

Scalable Persistent Storage in Dragonflow

- A pluggable Key-Value Interface Layer
- Supported Solutions
  - ETCD
  - RAMCloud
  - ZooKeeper
  - Redis
  - RethinkDB

Is it enough?
Scalable and reliable?

DB Consistency: Common Problem to all SDN Solutions



DB Consistency: Common Problem to all SDN Solutions

Neutron DB

- Relational Database
- ACID system
- Stores the whole virtualized network topology for OpenStack

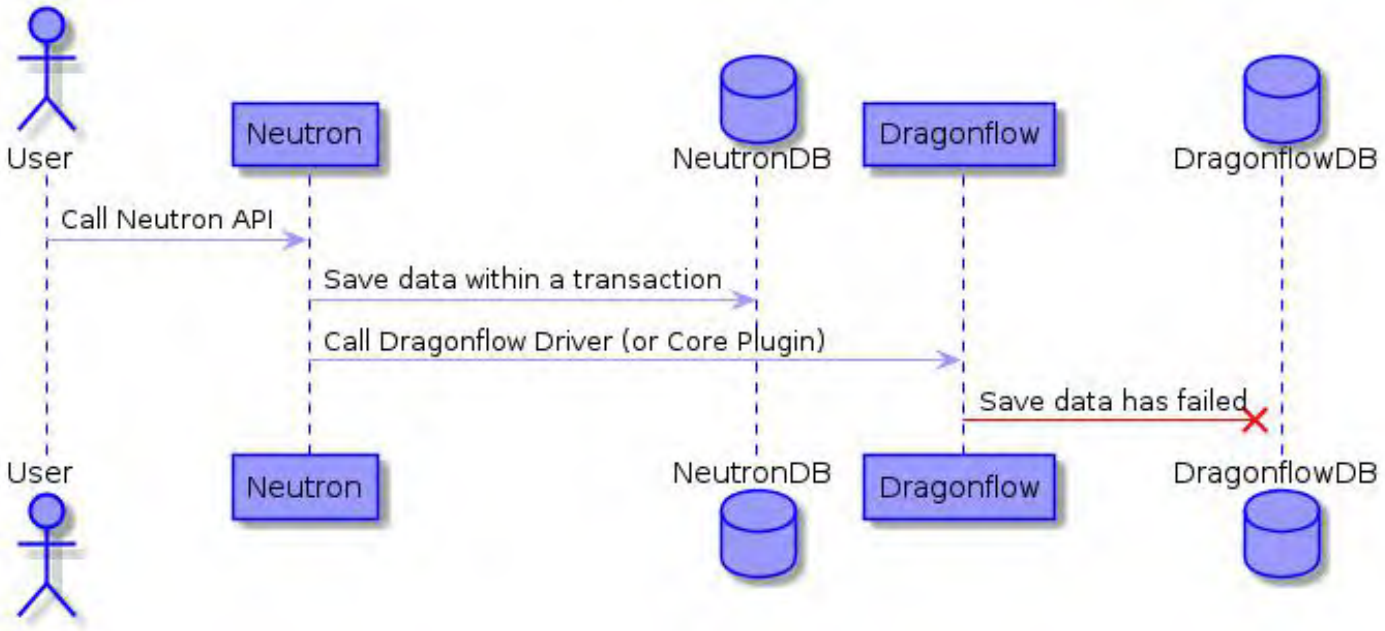
Dragonflow DB

- Key-value Store
- BASE system
- Stores a ‘partial’ virtualized network topology used in Dragonflow

DB Consistency: Common Problem to all SDN Solution

Problem 1: Dragonflow DB operation has failed

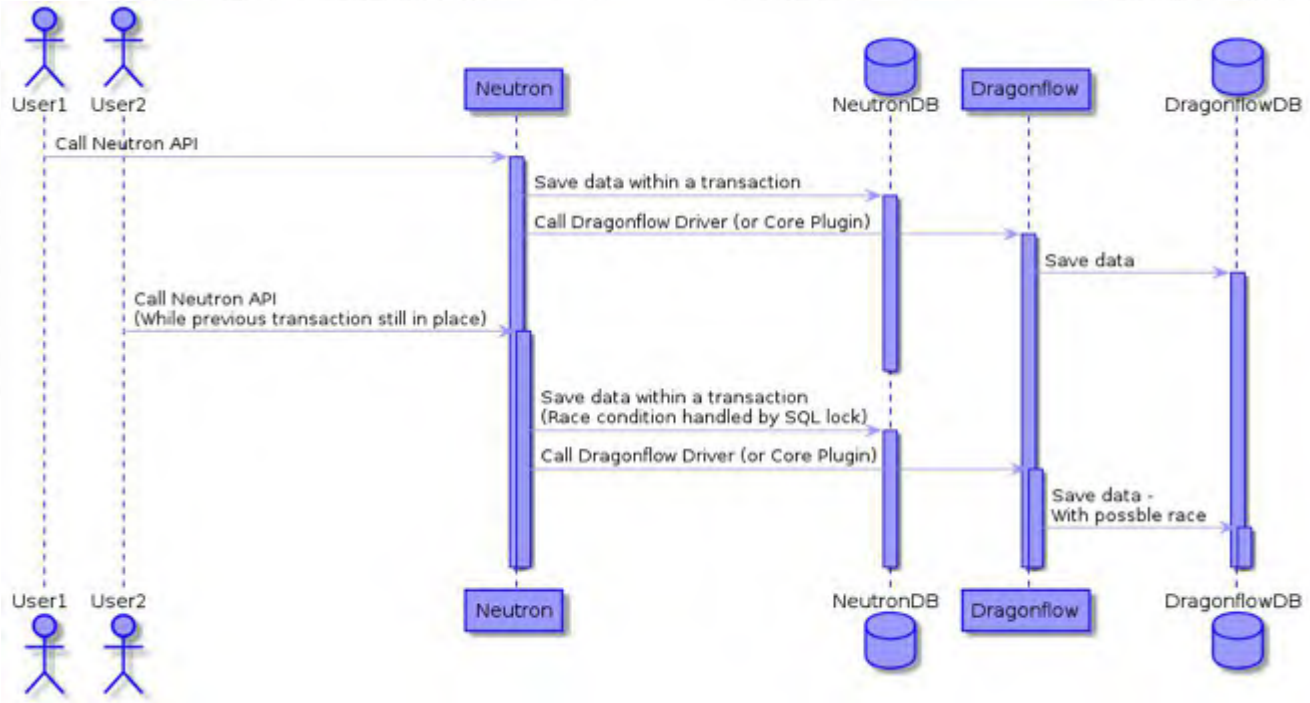
- Neutron DB operation is committed
- But the related Dragonflow DB operations have failed



DB Consistency: Common Problem to all SDN Solution

Problem 2: Multiple Parallel Transactions

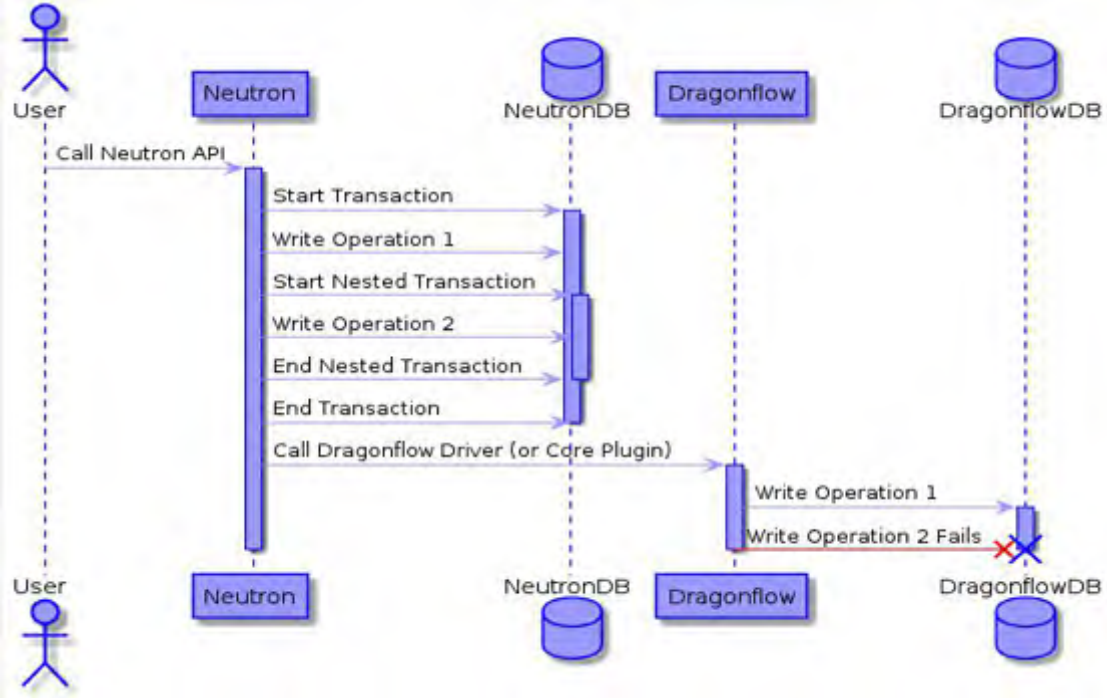
- Neutron DB can deal with multiple parallel transactions.
- How about Dragonflow DB?



DB Consistency: Common Problem to all SDN Solution

Problem 3: Nested Transactions

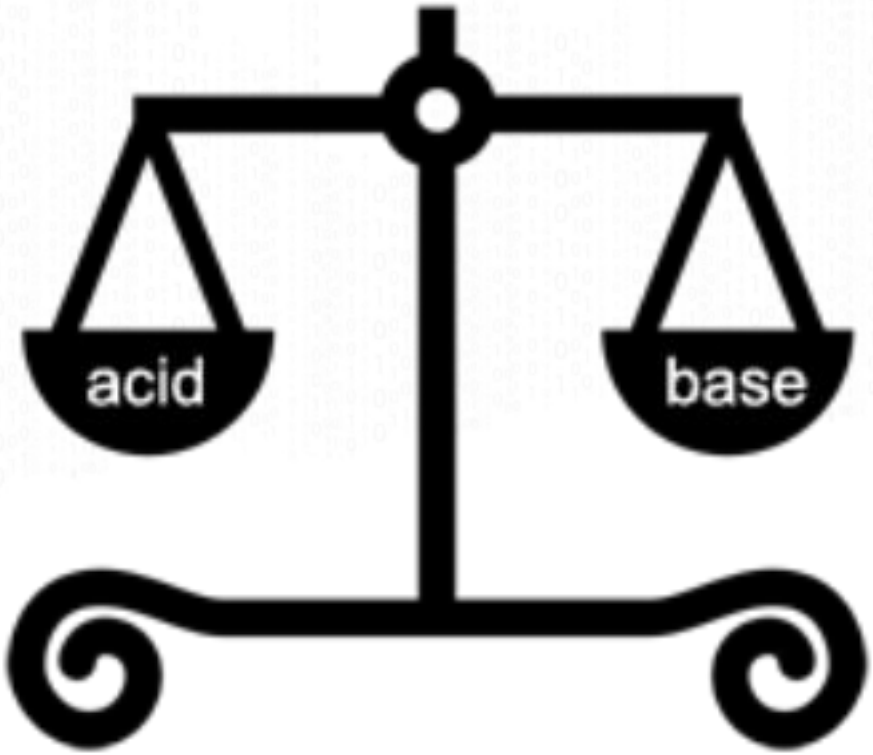
- Neutron DB can deal with nested transactions.
- How about Dragonflow DB?



DB Consistency: Common Problem to all SDN Solution

Additional Problems

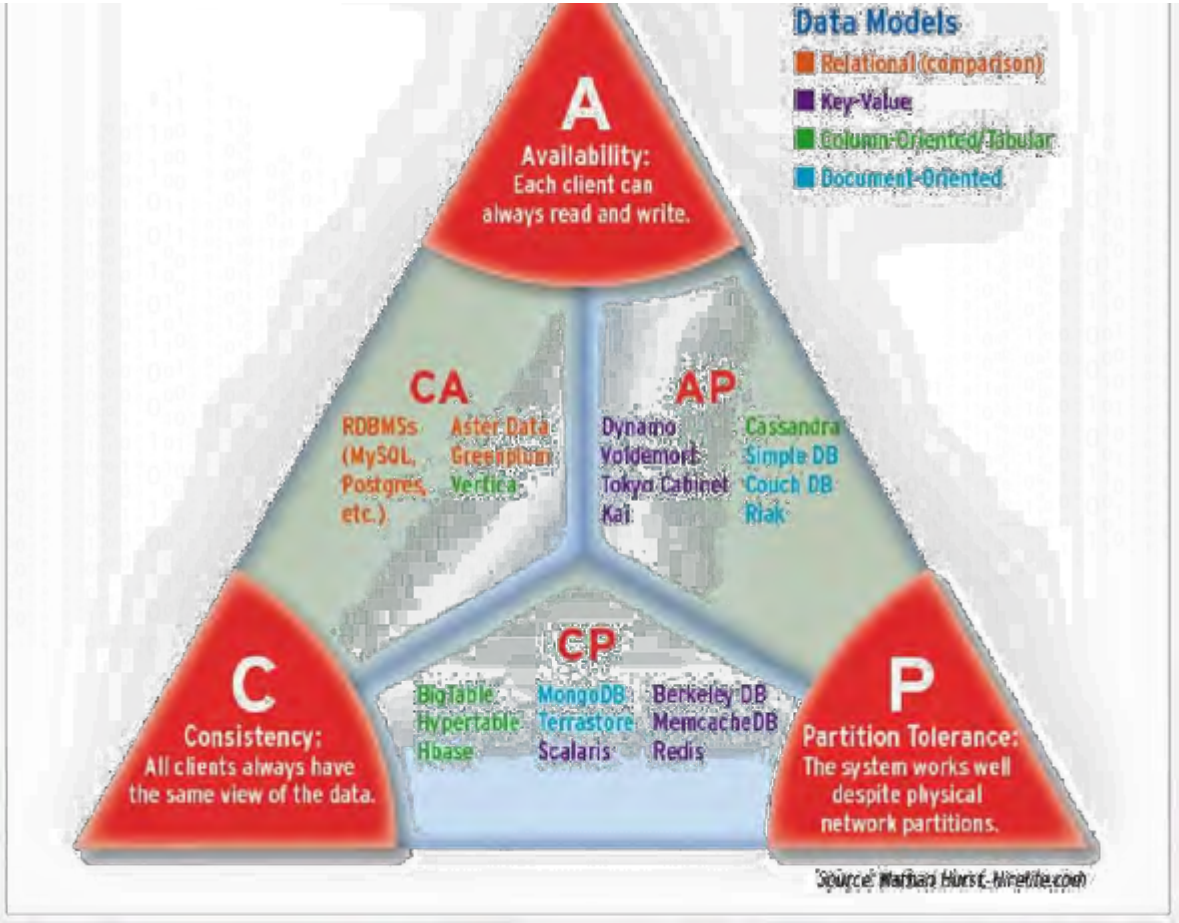
- There may be other issues.



Some thoughts on DB Consistency

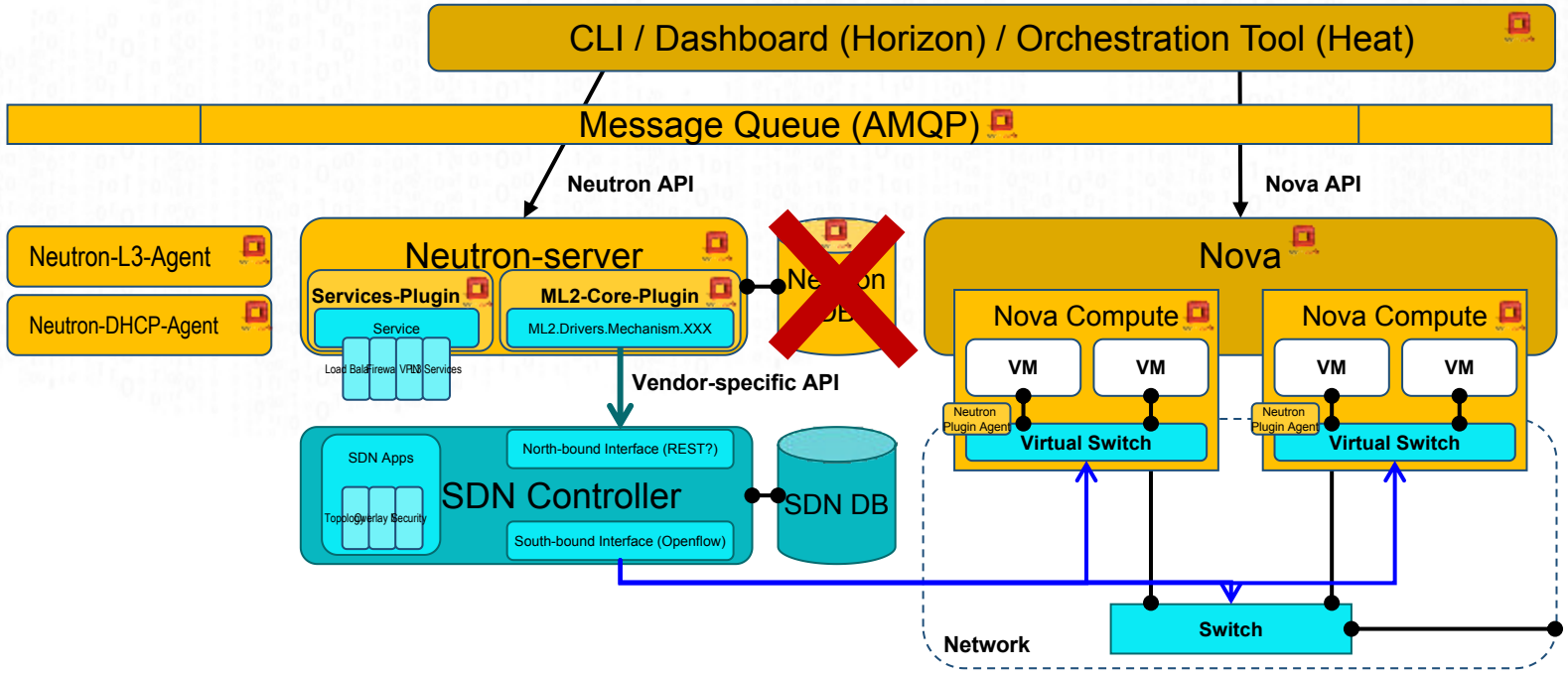
- Database in Multi-node/Multi-core System
  - Multi-Version Concurrency Control
  - Transaction Isolation
    - REPEATABLE READ
    - READ COMMITTED
    - READ UNCOMMITTED
    - SERIALIZABLE

Some thoughts on DB Consistency



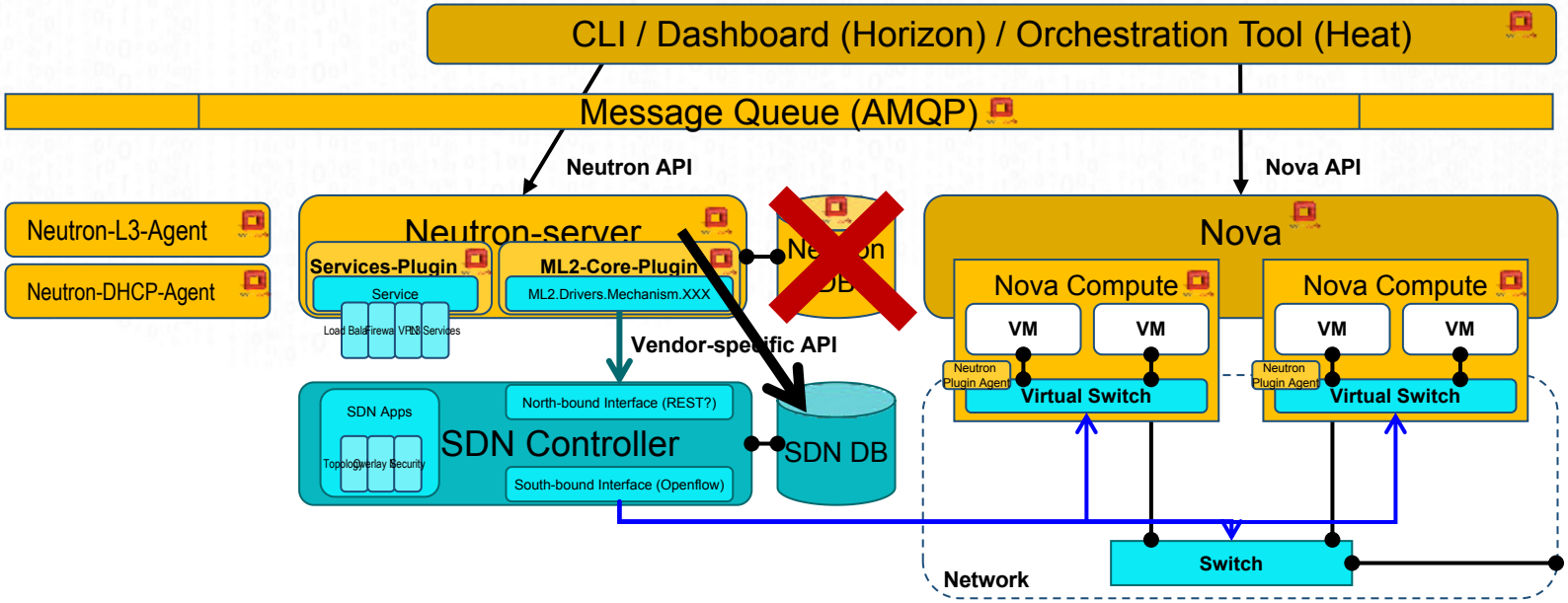
Some thoughts on DB Consistency

- Remove Neutron DB
  - Complicated Solution when involving ML2
  - Cannot be done in a short period of time



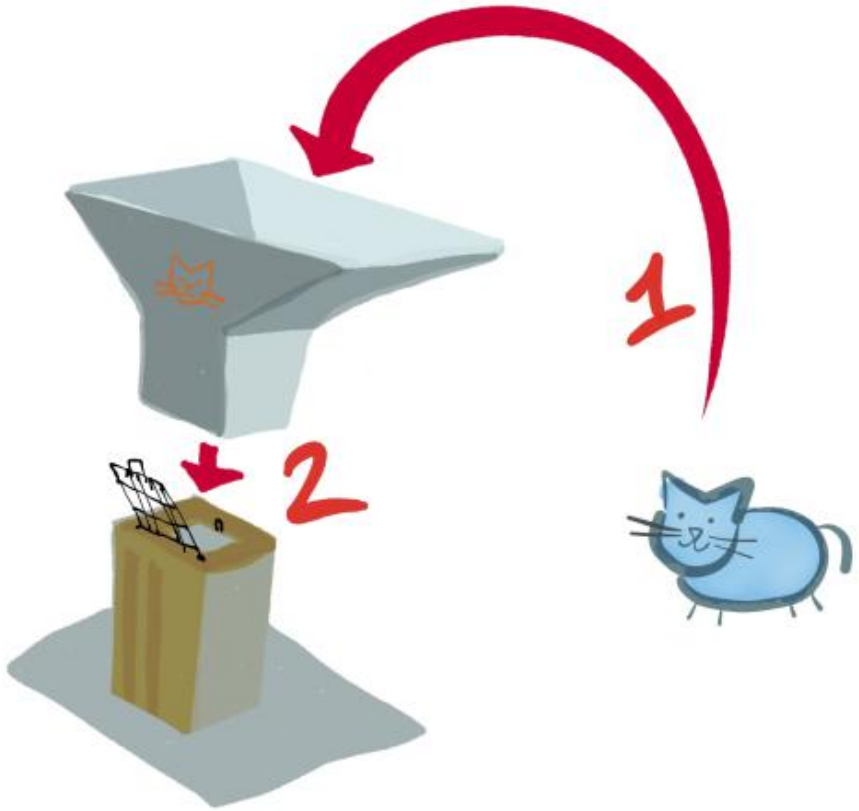
Some thoughts on DB Consistency

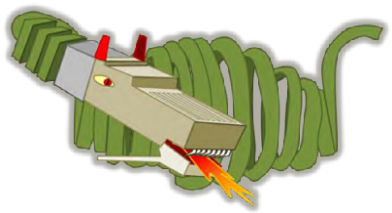
- Introduce the pluggable key-value store into Neutron
  - How to work with SQLAlchemy?
    - ROME: https://github.com/BeyondTheClouds/rome
  - Need much more time on evaluation and deep discussion.



Some thoughts on DB Consistency

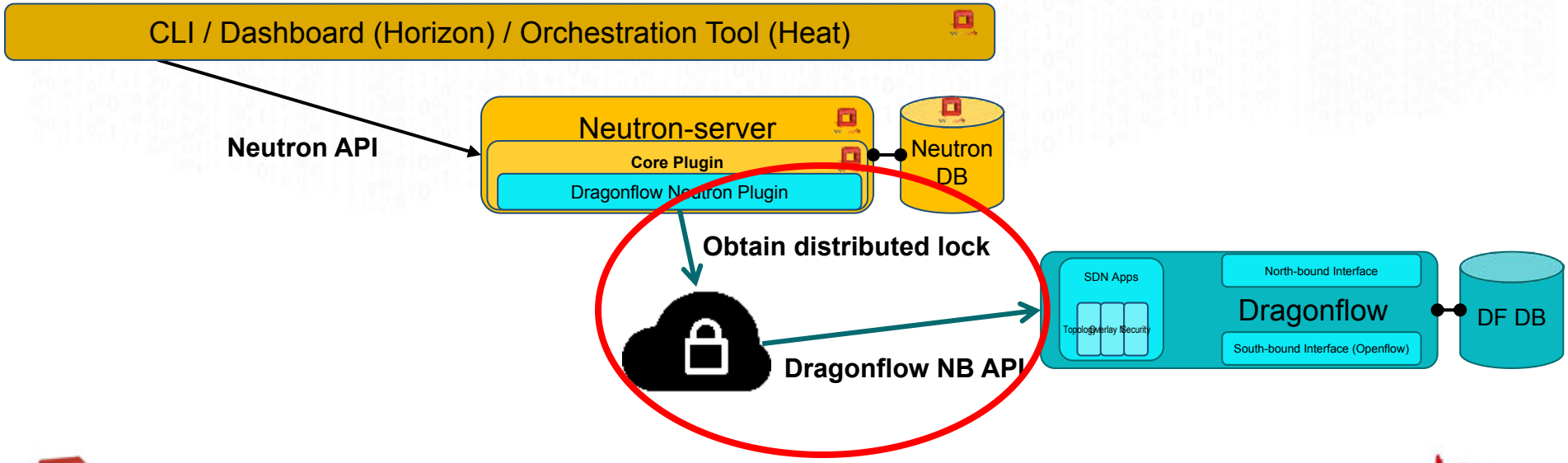
- Are there any other solutions?
  - That are simple?
  - That are straightforward?

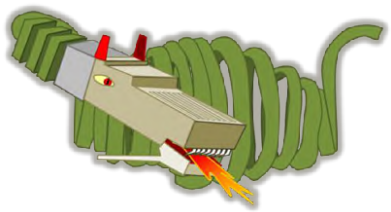




DB Consistency in Dragonflow —— Distributed Lock

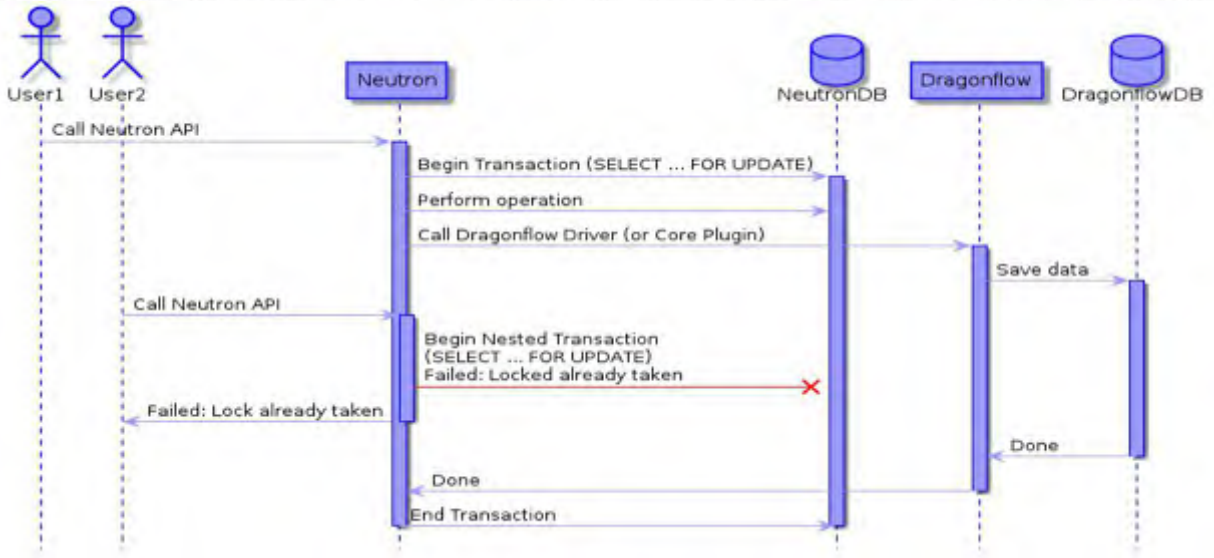
- Introduce a distributed lock for coordination
  - Guarantee the atomicity of a given API
  - Implemented in the Neutron core plugin layer
  - Project-based lock allows concurrency

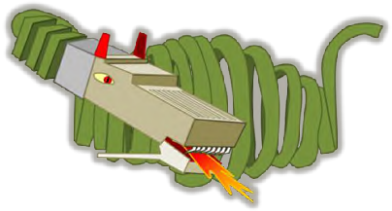




DB Consistency in Dragonflow —— Distributed Lock

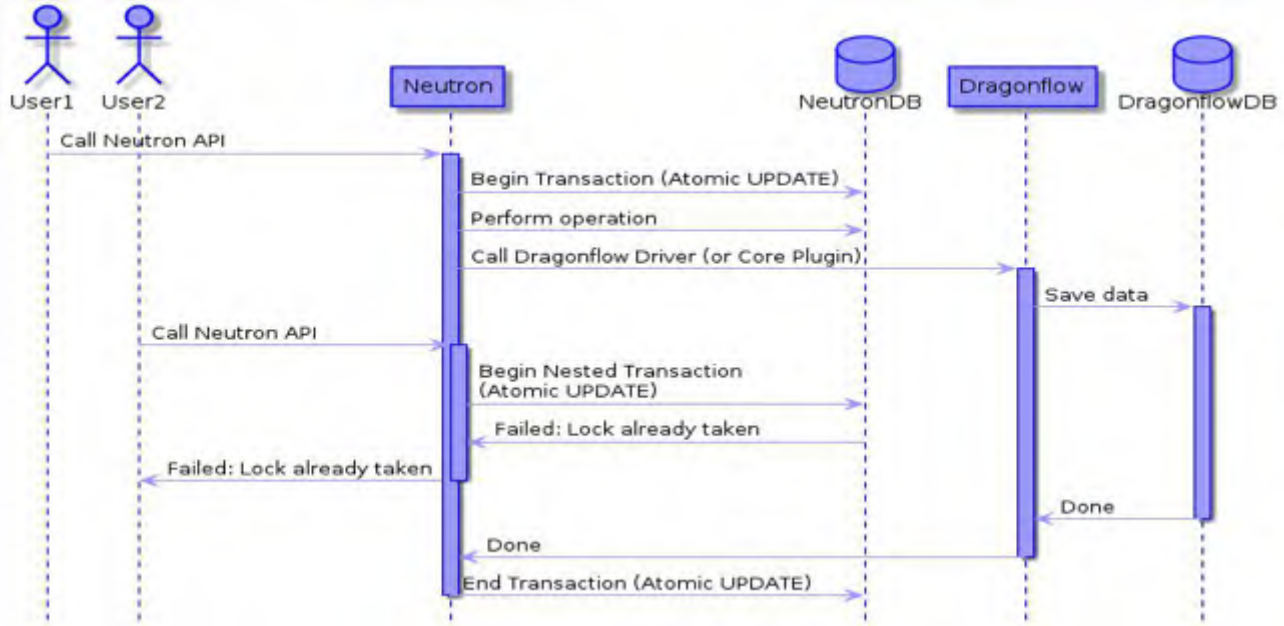
- Initial Solution: Introduce a distributed lock for coordination
  - Initially it was implemented by SELECT-FOR-UPDATE statement
  - Not compatible with Galera clustering
  - Performance penalty when involving retry_for_deadlock operation

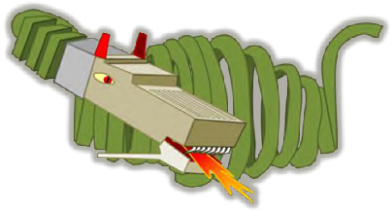




DB Consistency in Dragonflow —— Distributed Lock

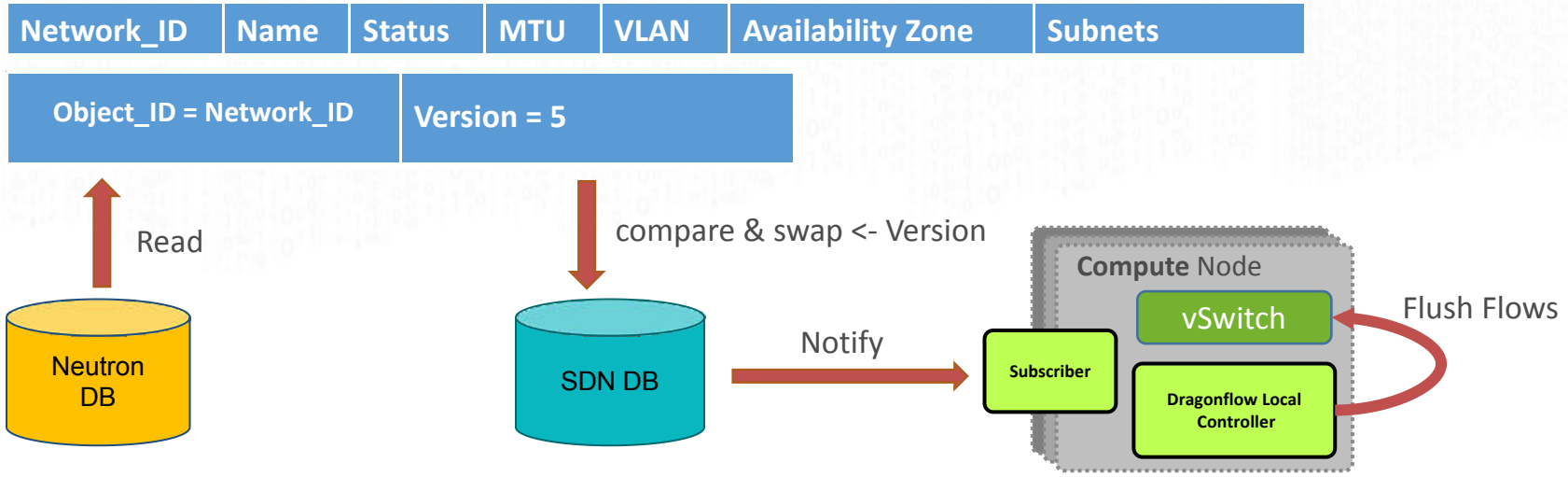
- Improved Solution: SQL-based compare-and-swap operation
  - Compatible with Galera clustering
  - No performance penalty

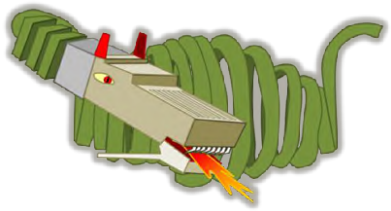




DB Consistency in Dragonflow —— Object Synchronization

- Introduce an object synchronization mechanism
  - All the objects stored in both databases are versioned.
  - Sync the object when something unexpected happens.

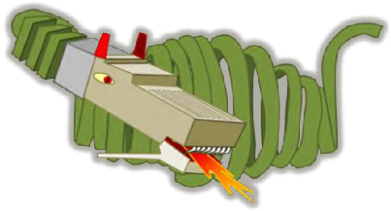




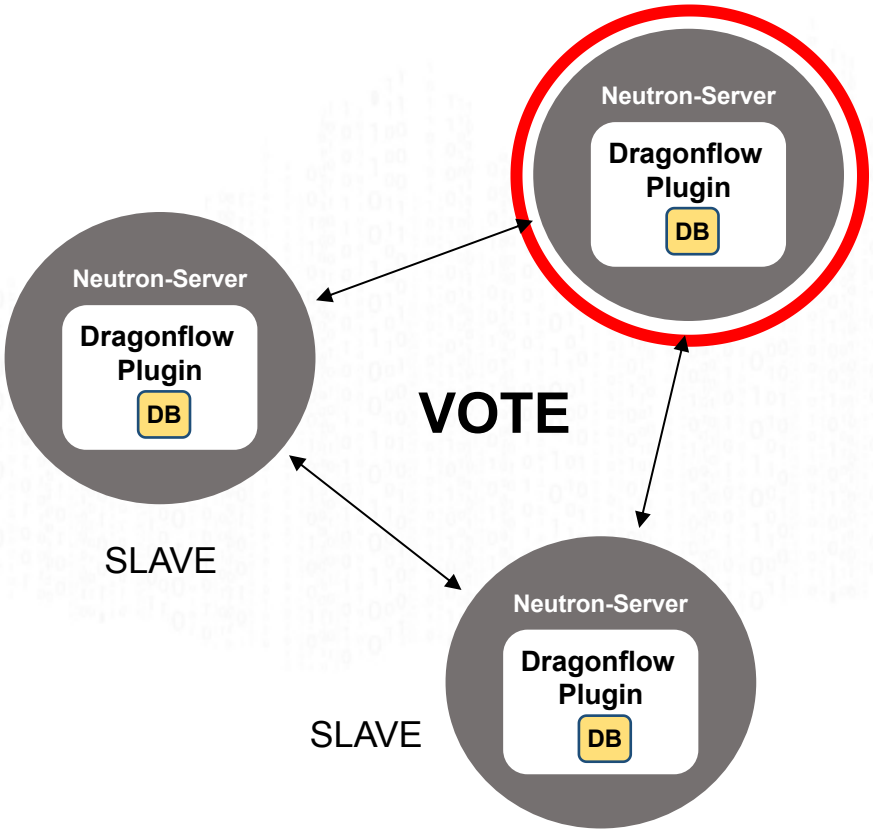
DB Consistency in Dragonflow —— Auto-Recovery

- Introduce auto-recovery mechanism
  - Periodically detect inconsistency by version comparison.
  - Recover the object data from Neutron DB to Dragonflow DB.
  - Compatible for multi-node deployment.
    - Introduce Master Election
    - Introduce Load Balancing in the later phase

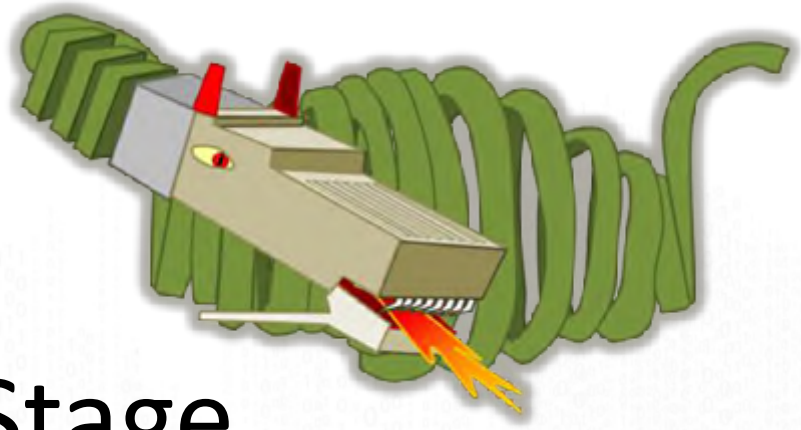
OPENSTACK DAYS CHINA



DB Consistency in Dragonflow —— Auto-Recovery



1	Obtain the distributed lock
2	Get Object data from Neutron DB
3	Get the corresponding data from Dragonflow DB
4	Verify the consistency of that data
5	Call update_object for that data if it is not consistent
6	Check the next object.
7	Release the distributed lock



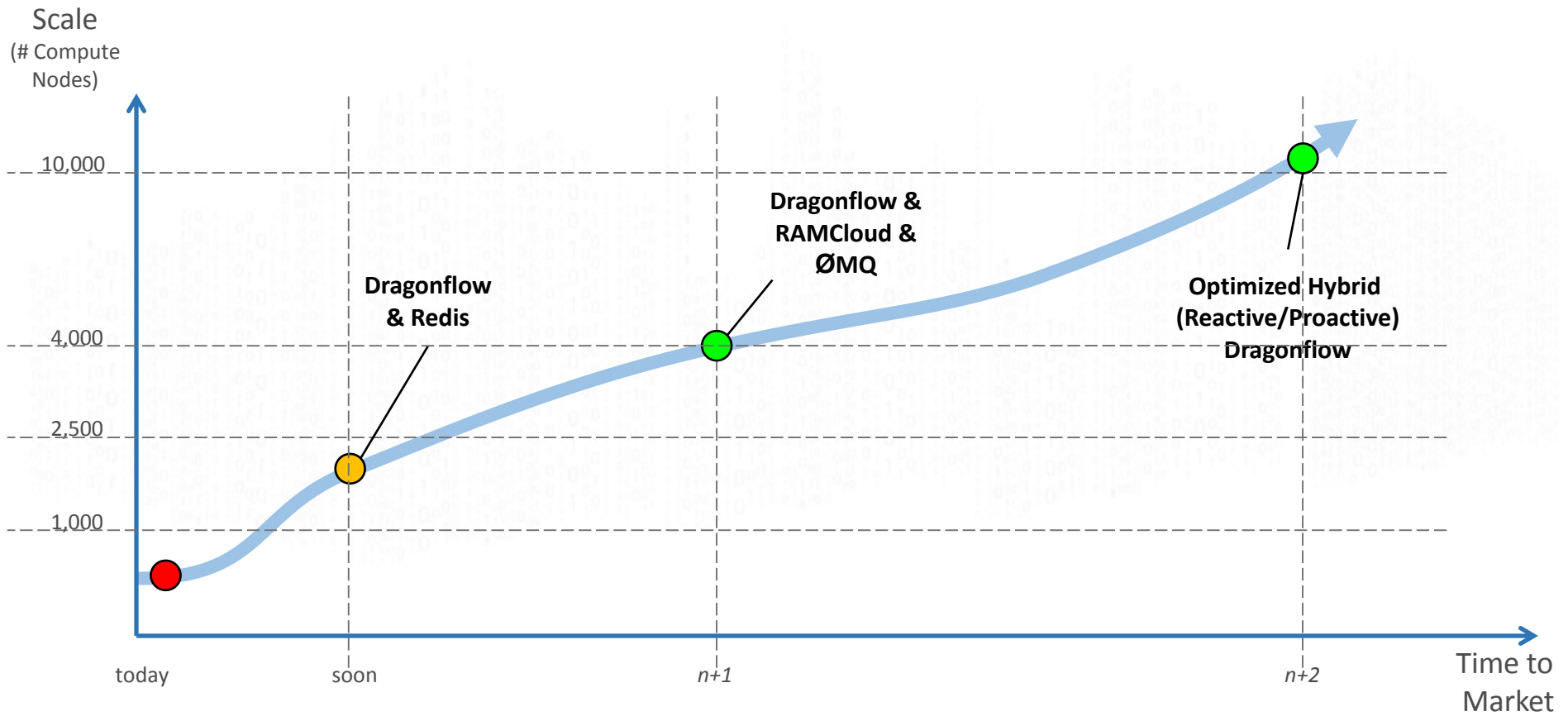
To the Next Stage

OPENSTACK DAYS CHINA

OpenStack Challenges

- **Scalability**
  - Networking does not scale (< 500 compute nodes)
- **Performance**
  - Networking performance is low (namespace overhead, huge control plane overhead, …)
- **Operability**
  - Reference implementation has lots of maintenance problems (e.g. thousands of concurrent DHCP servers, namespaces, etc.)

Scalability



Roadmap

- Additional DB Drivers ZooKeeper, Redis…
- Selective Proactive DB
- Pluggable Pub/Sub Mechanism
- DB Consistency
- Distributed DNAT
- Security Group

- Hierarchical Port Binding (SDN ToR) move to ML2
- Containers (Kuryr plugin and nested VM support)
- Topology Service Injection / Service Chaining
- Inter Cloud Connectivity (Border Gateway / L2GW)
- Optimize Scale and Performance

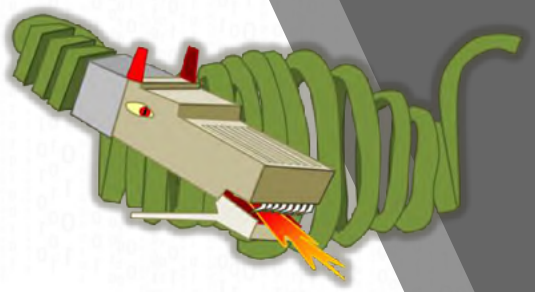
OPENSTACK DAYS CHINA

Newton Release New Applications

- IGMP Application
- Distributed Load Balancing (East/West)
- Brute Force prevention
- DNS service
- Distributed Metadata proxy
- Port Fault Detection

OPENSTACK DAYS CHINA

Ride the Dragon!



- Documentation
  - https://wiki.openstack.org/wiki/Dragonflow
- Bugs & blueprints
  - https://launchpad.net/dragonflow
- DF IRC channel
  - #openstack-dragonflow
  - Weekly on Monday at 0900 UTC in #openstack-**meeting**-4 (**IRC**)

Thanks