



**CLOUD NATIVE**  
COMPUTING FOUNDATION

# Nezha: A Kubernetes Native Big Data Accelerator For Machine Learning

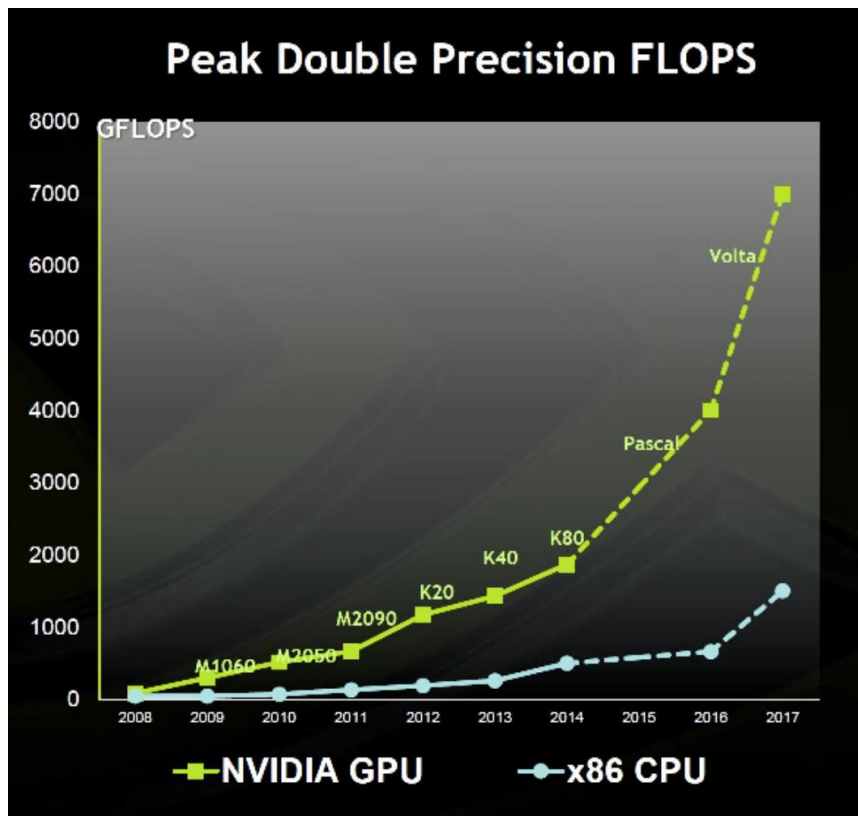
Huamin Chen - Red Hat  
Yuan Zhou - Intel  
Dec, 2018

- Background and Motivations
- Architecture Overview
- Use Cases in kubeflow (Deep Learning) and Spark-SQL (Data Analytics)
- Summary & Next Step

# Deep Learning Accelerators

- Hardware Accelerator
  - CPU, GPU, TPU, FPGA
- Software Accelerator
  - Better Algorithms
  - In-memory Software Stack

# While Compute Is Accelerated



Source: <https://nar.ucar.edu/2017/ral/gpu-accelerated-microscale-modeling-fasteddy>

# Data Link Is Lagging

Dataset Name	Size	Download Time	Training Time (with GPU)
MNIST	10+M	0.5~2 minutes	minutes
CIFAR 10/100	100+M	2~30 minutes	minutes
COCO 2017	26G	2~Many hours	hours~days

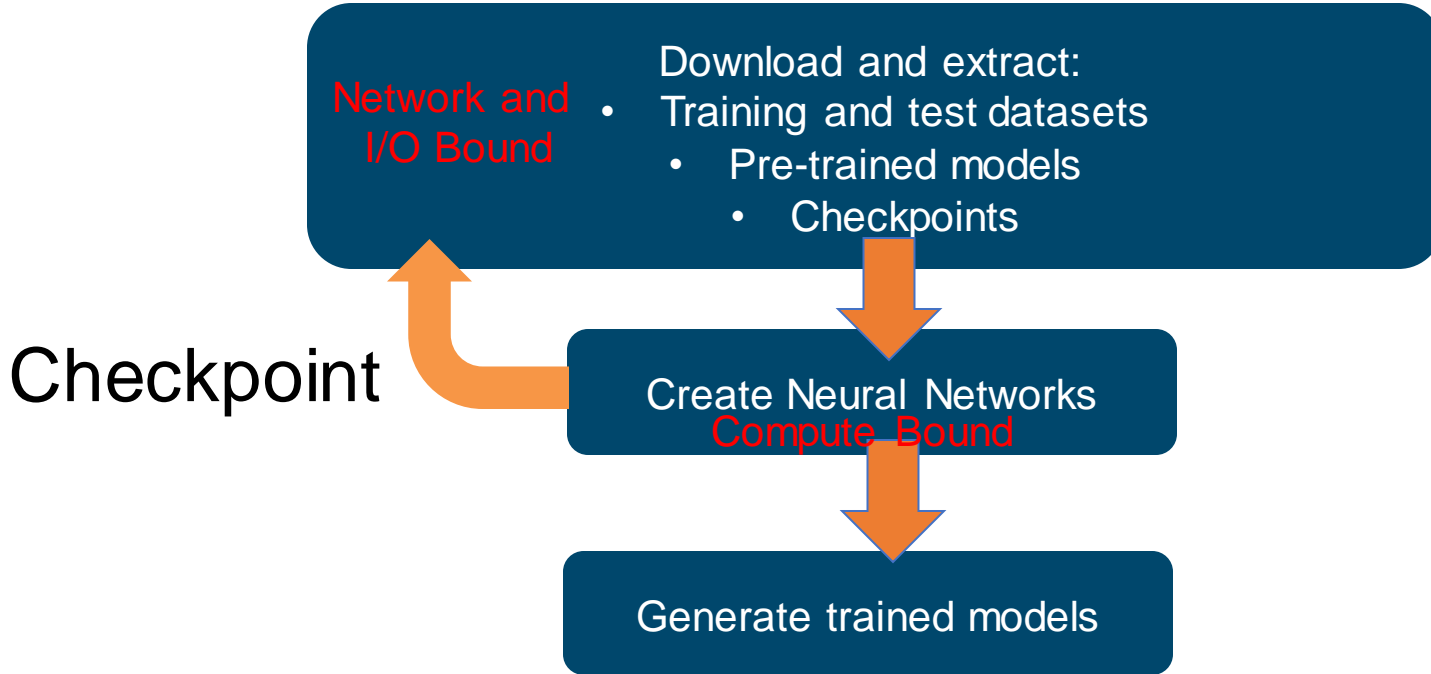
# How to Accelerate Data Link?

Solution	Comment
Build a customized filesystem	<ul style="list-style-type: none"><li>• Take years to succeed</li><li>• Very configuration dependent</li></ul>
Cache Dataset on Compute node	At the expense of cost, flexibility, and mobility
Nezha: On-demand cache	Kubernetes native, run everywhere, application transparent

# Project Goals

- Nezha aims to accelerate large dataset download and extract to keep input pipeline at high rate.
  - Network : Reverse Proxy Cache
  - I/O: Bring data close to compute
- Kubernetes Native
  - Services, Deployments, HostAlias, Persistent Volumes, and WebHook.
- *(Almost) transparent to end users!*
  - *Though TLS certificate issue requires some head scratching...*

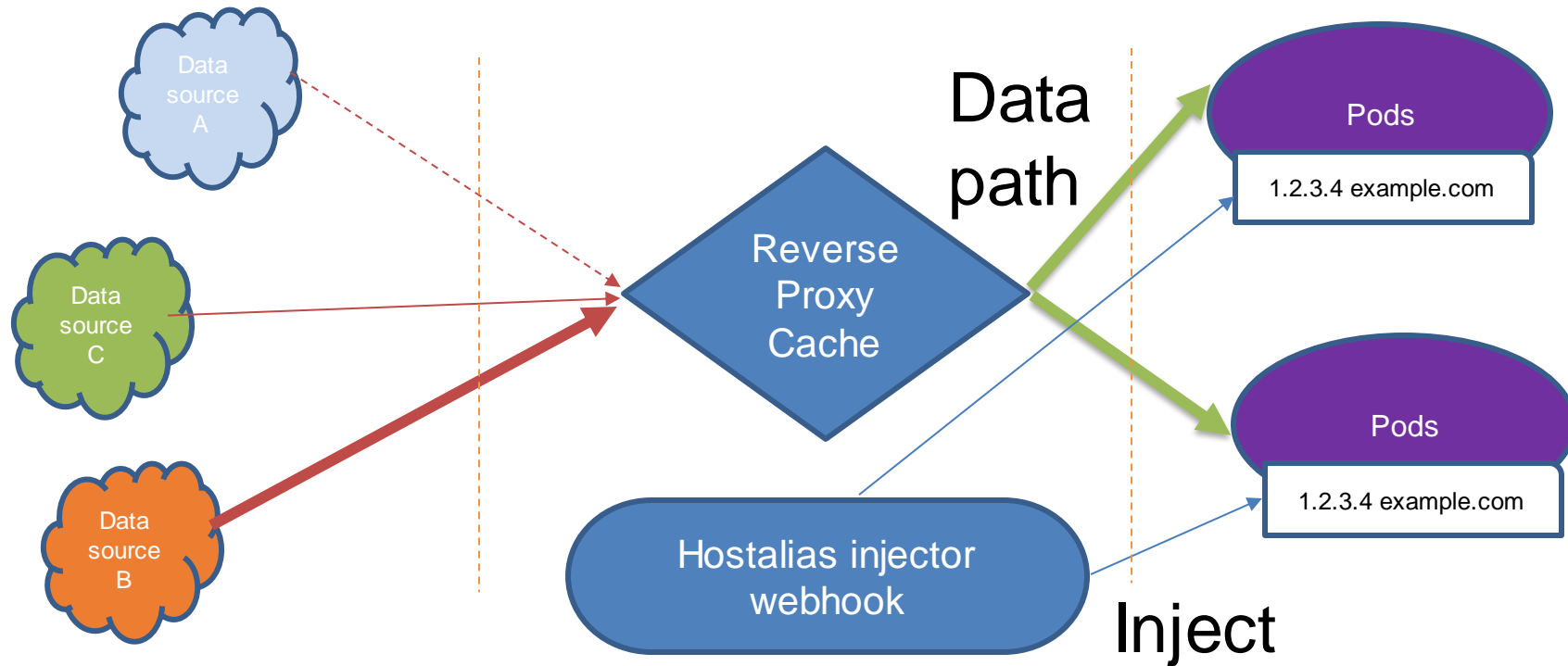
# A Walk of Deep Learning Training





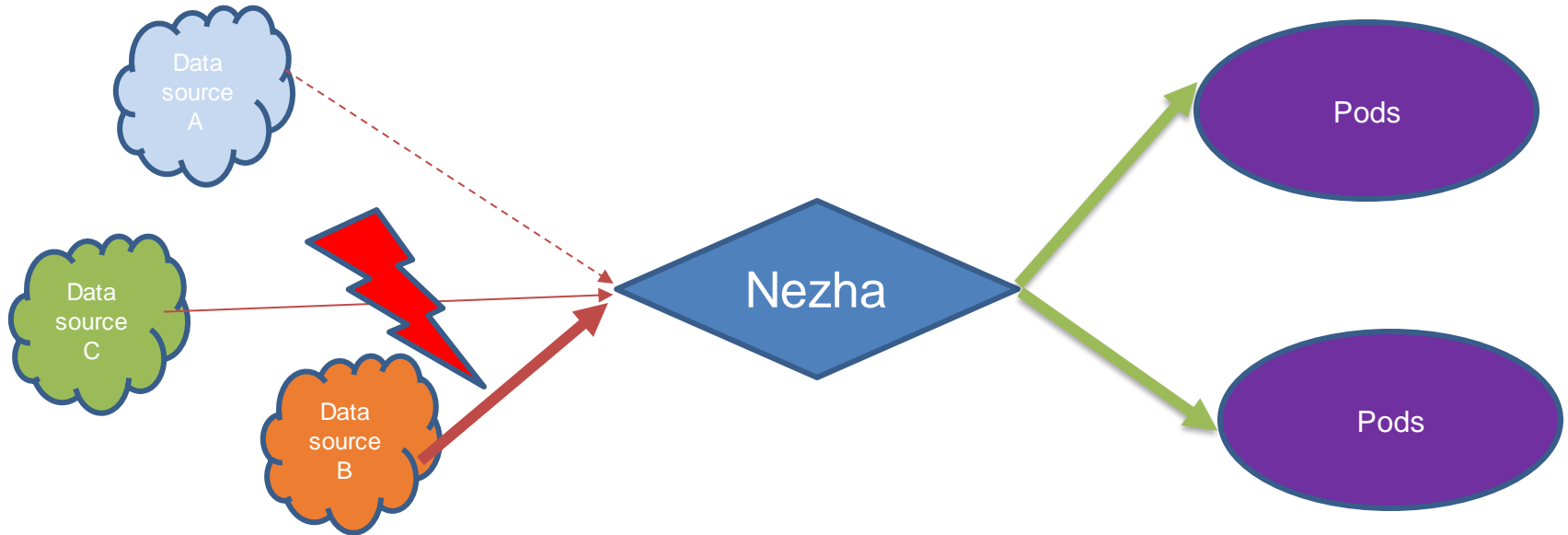
# Nezha Architecture

<https://github.com/fast-m1/nezha>

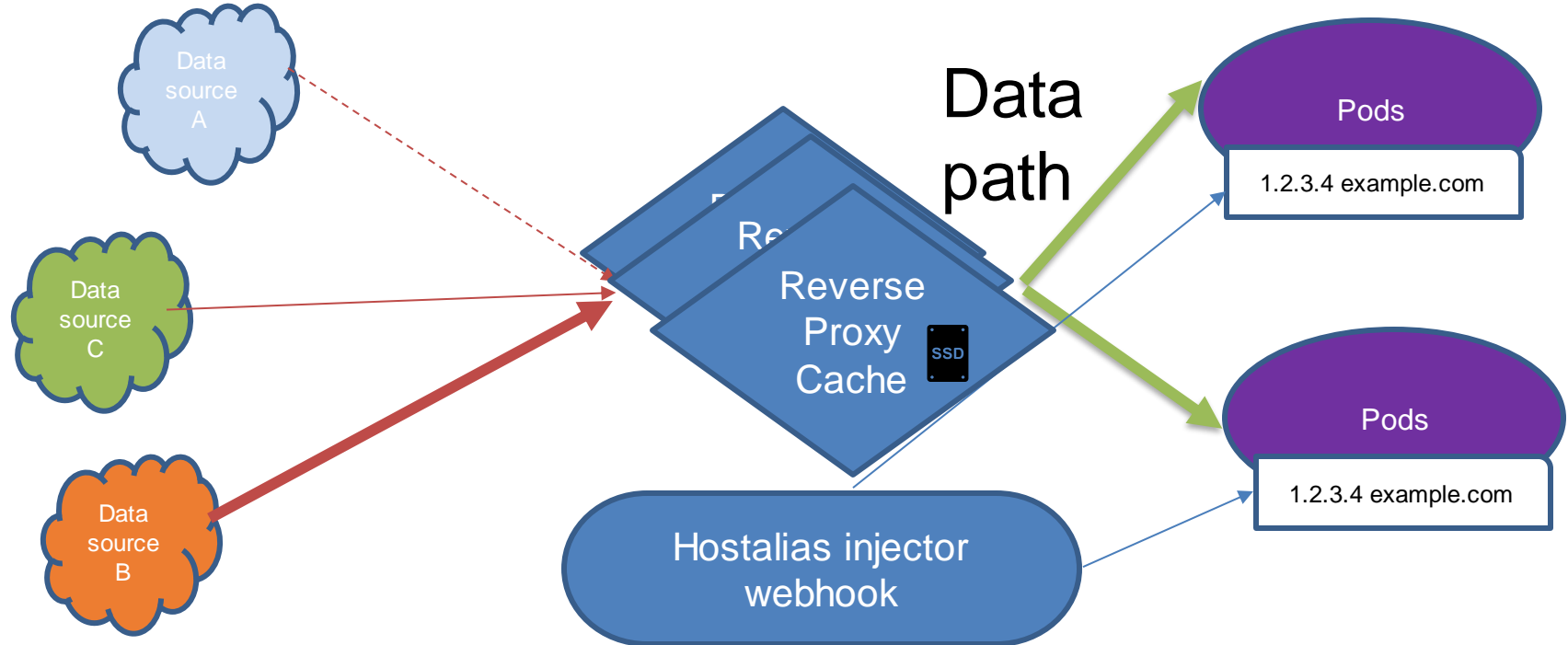


# Nezha: No More Data Silos

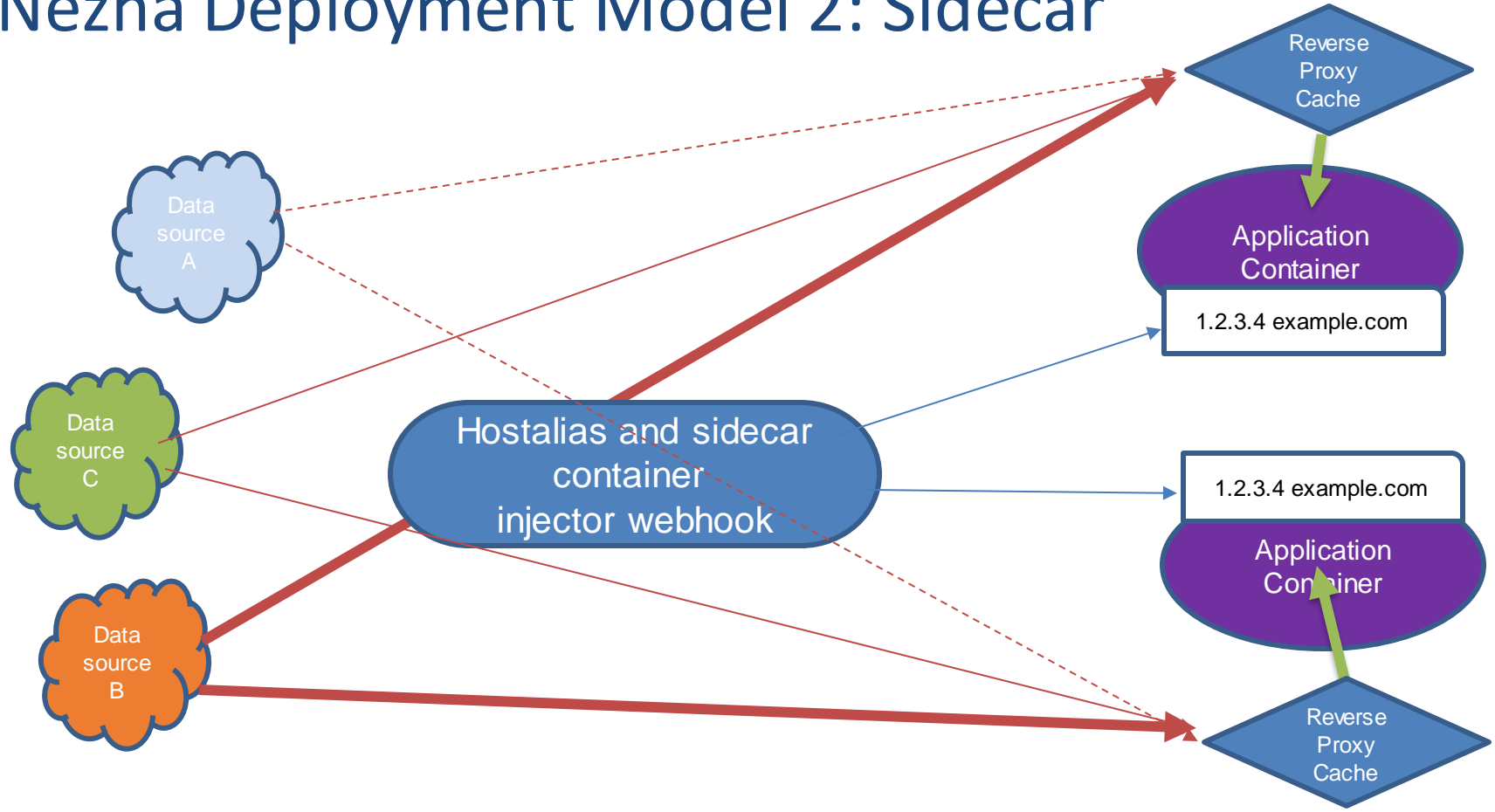
- Nezha aggregates data sources, brings consistent I/O performance and high availability, without big price ticket or proprietary technologies.



# Nezha Deployment Model 1: Centralized



# Nezha Deployment Model 2: Sidecar



# Deployment Comparision

	Non Nezha	Nezha: Centralized	Nezha: Sidecar
Network Performance	Subject to <b>external</b> network performance	Subject to <b>internal</b> network performance	Subject to <b>local container</b> network performance
Data Availability	<b>All</b> compute tasks are subject to data source and network availability	<b>All</b> compute tasks are subject to proxy container availability	<b>Individual</b> compute task is subject to its sidecar container availability

# Deploy Nezha in Kubernetes

- Instructions can be found at github repo
- Get to know what labels the applications use
- Kubeflow job deployed by ksonnet has the following signature

```
apiVersion: batch/v1
kind: Job
metadata:
  labels:
    app.kubernetes.io/deploy-manager: ksonnet
```

# Configuration in the ConfigMap

```
apiVersion: v1
kind: ConfigMap
data:
  config: |
    - name: dataset
      app: app.kubernetes.io/deploy-manager
      label: ksonnet
      hostAliases:
        - ip: "10.106.122.174"
          hostnames:
            - "download.tensorflow.org"
            - "yann.lecun.com"
            - "www.cs.toronto.edu"
            - "storage.googleapis.com"
            - "images.cocodataset.org"
```

Kubeflow label



Proxy service IP



Upstream servers



# Hostaliases Injection in Action

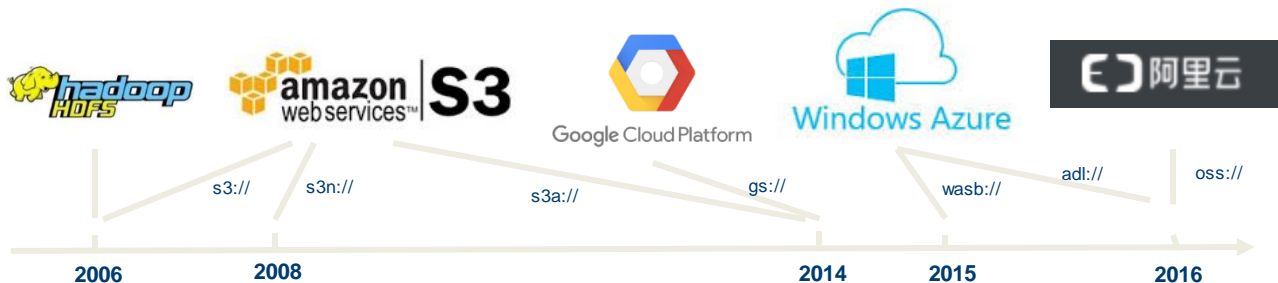
```
spec:
  containers:
    - args:
      - http://images.cocodataset.org/zips/train2017.zip
      image: mwendler/wget
      imagePullPolicy: Always
      name: wget
  hostAliases:
    - hostnames:
      - download.tensorflow.org
      - yann.lecun.com
      - www.cs.toronto.edu
      - storage.googleapis.com
      - images.cocodataset.org
      ip: 10.106.122.174
```



# Bigdata analytics are moving to cloud



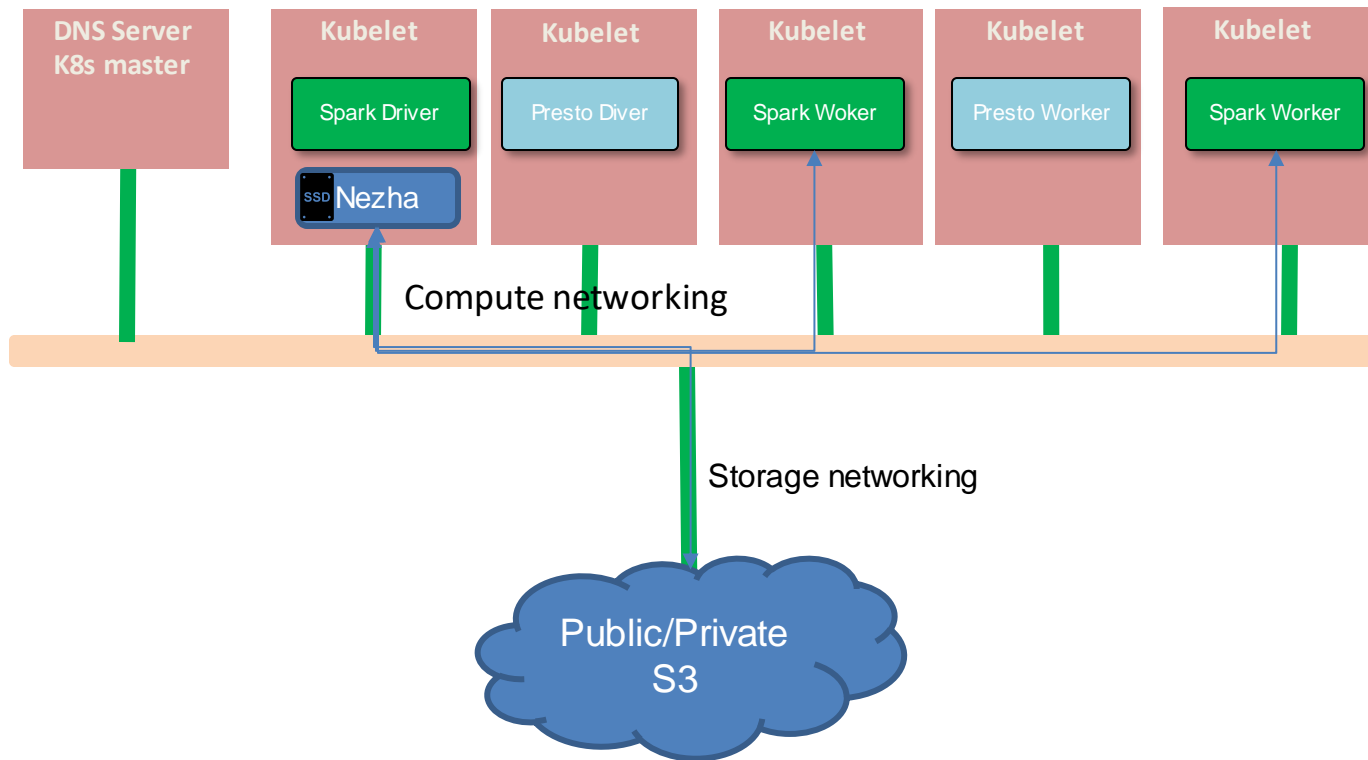
Hadoop Compatible File System abstraction layer: Unified storage API interface Hadoop fs -ls s3a://job/



# Hadoop object store connector

- Bigdata can now talk to public object stores directly
  - S3a: Amazon Simple Storage Service
  - Wasb: Windows Azure Storage Blob
  - Adl: Azure Data Lake Store
  - Gs: Google Cloud Storage
- Usually data reading is slower due to separate storage networking
- Pay on multiple get requests if using public S3

# Example Nezha Deployment For S3 caching



# Intel Optane DC Persistent Memory



Big and Affordable Memory

128, 256, 512GB

High Performance Storage

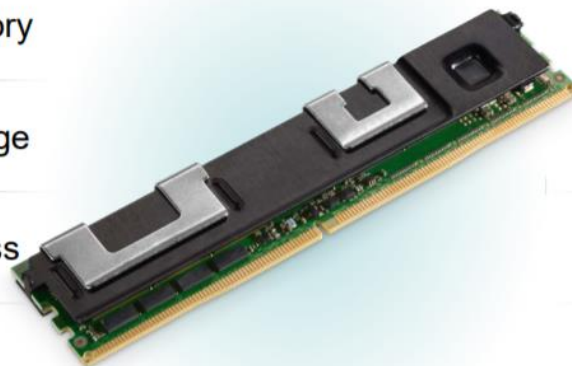
DDR4 Pin Compatible

Direct Load/Store Access

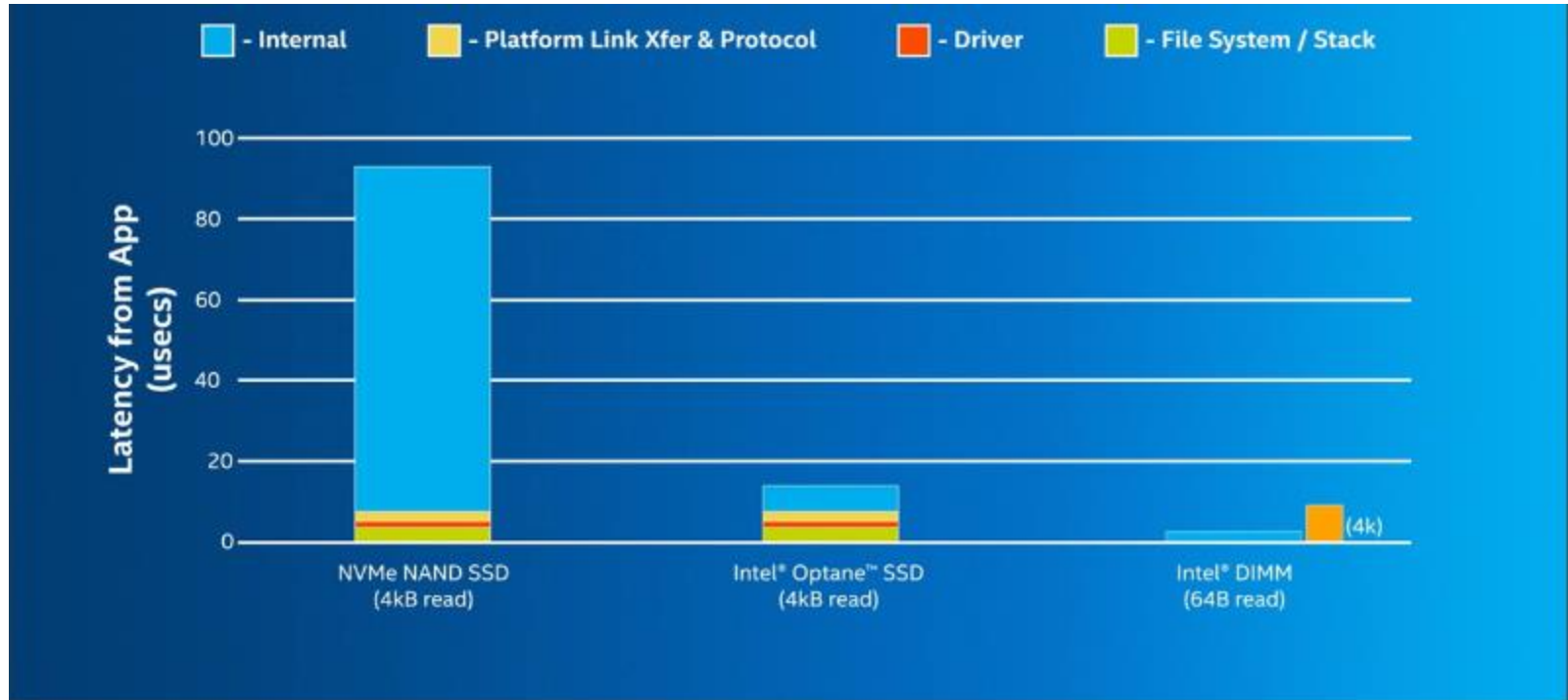
Hardware Encryption

Native Persistence

High Reliability



# Intel DCPMM Application Latency



# Kubernetes CSI plugin for Intel DCPMM

- A new CSI plugin for Intel DCPMM
  - Dynamic provisioning of node-local AEP memory in Kubernetes as block devices, as specified by Kubernetes CSI specification
- Working on this now!
  - <https://github.com/intel/pmem-CSI>
  - Go to Intel booth for more details

# Summary & Next Steps

- Nezha is a Kubernetes Native Big Data Accelerator to improve the input data performance
- HostAlias and webhook used thus it's transparent for application
- Applies to ML/DL workloads, as well as bigdata analytics based on object storage
- Next steps
  - HTTPS/TLS support
  - Explore I/O acceleration via Sidecar container + Burst Buffer