KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

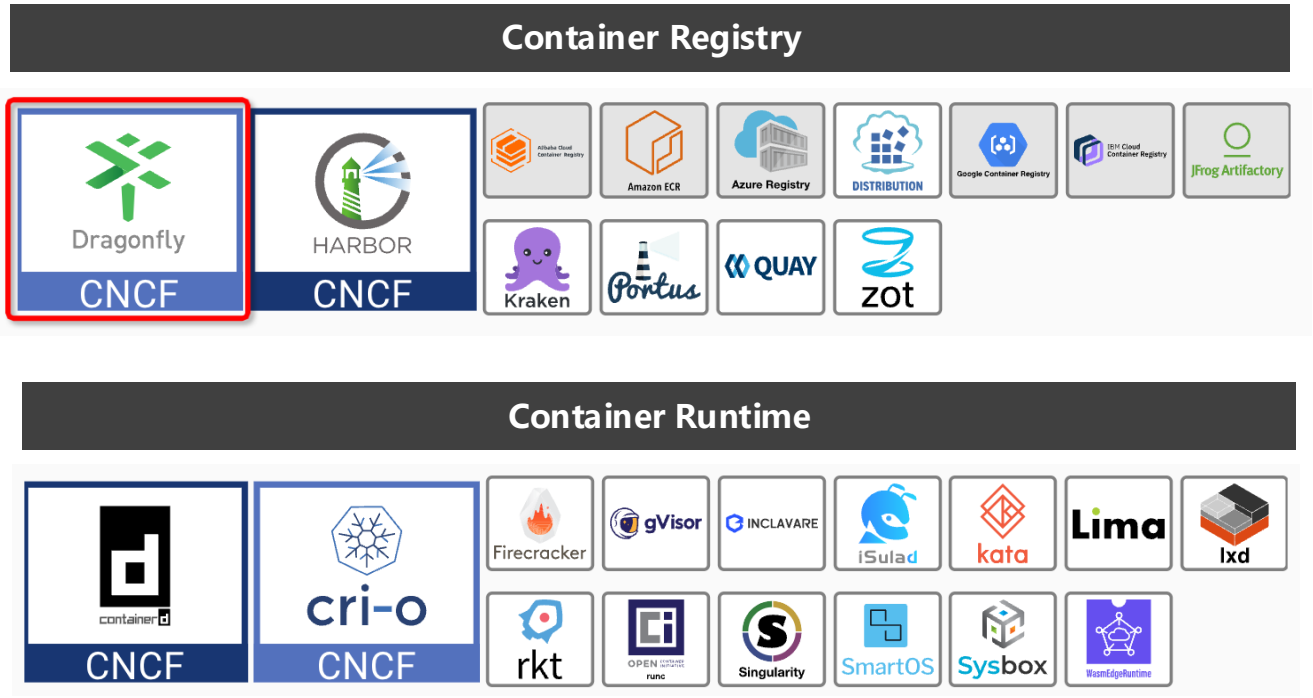
China 2024

Dragonfly: Intro, Updates and Ant Group's Practice of Accelerating Model Distribution in Ray Serving

Qixiang Chen @ AntGroup & Wenbo Qi(Gaius) @ AntGroup

Introduction

CLOUD NATIVE Landscape



Dragonfly

CLOUD NATIVE COMPUTING FOUNDATION

What is Dragonfly?

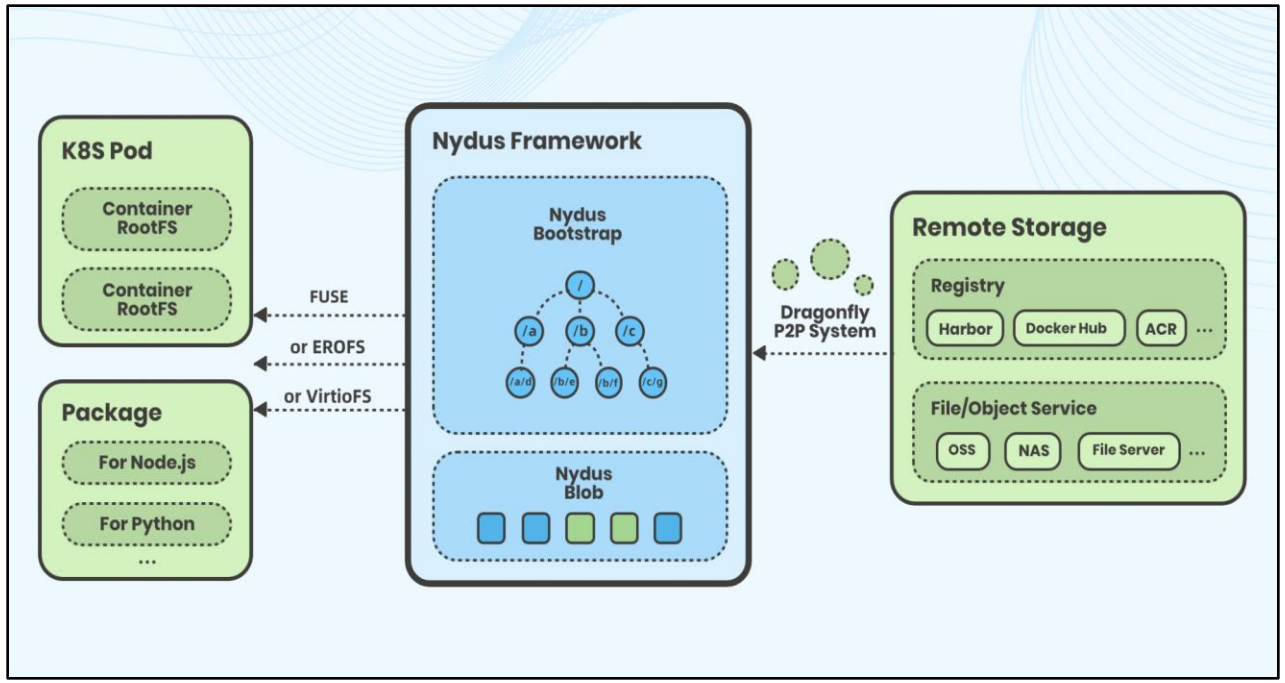
Provide efficient, stable and secure file distribution and image acceleration based on p2p technology to be the best practice and standard solution in cloud native architectures.

It is hosted by the Cloud Native Computing Foundation(CNCF)as an ***Incubating*** Level Project.

There are more than ***100*** contributors, and maintainers come from ***Ant Group***, ***Alibaba Group***, ***ByteDance***, ***Intel***, ***Baidu Group***, ***Zhipu AI*** and ***Dalian University of Technology***.

Public cloud users include ***Alibaba Cloud(Aliyun)***, ***Google Cloud Platform (GCP)***, ***Volcano Engine***, ***Baidu AI Cloud***, etc.

Introduction



What is Nydus?

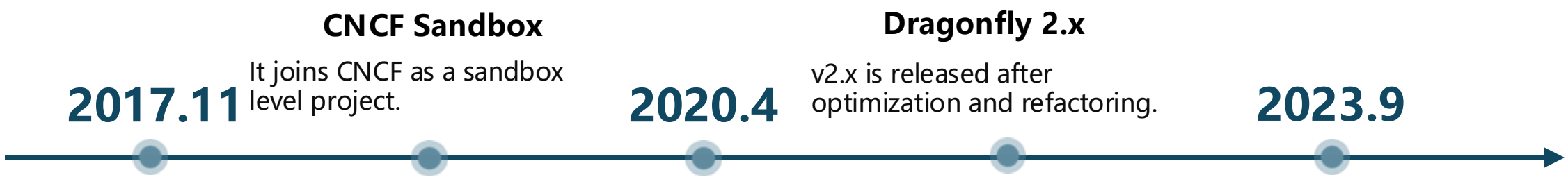
Provide a content-addressable file system on the RAFS format, which enhances the current OCI image specification by improving container launch speed, image space and network bandwidth efficiency, and data integrity.

It is a ***sub-project of Dragonfly***. It can reduce the end-to-end cold start time of containers from ***minutes to seconds***, and supports the creation of ***millions of containers*** every day in production.

NYDUS

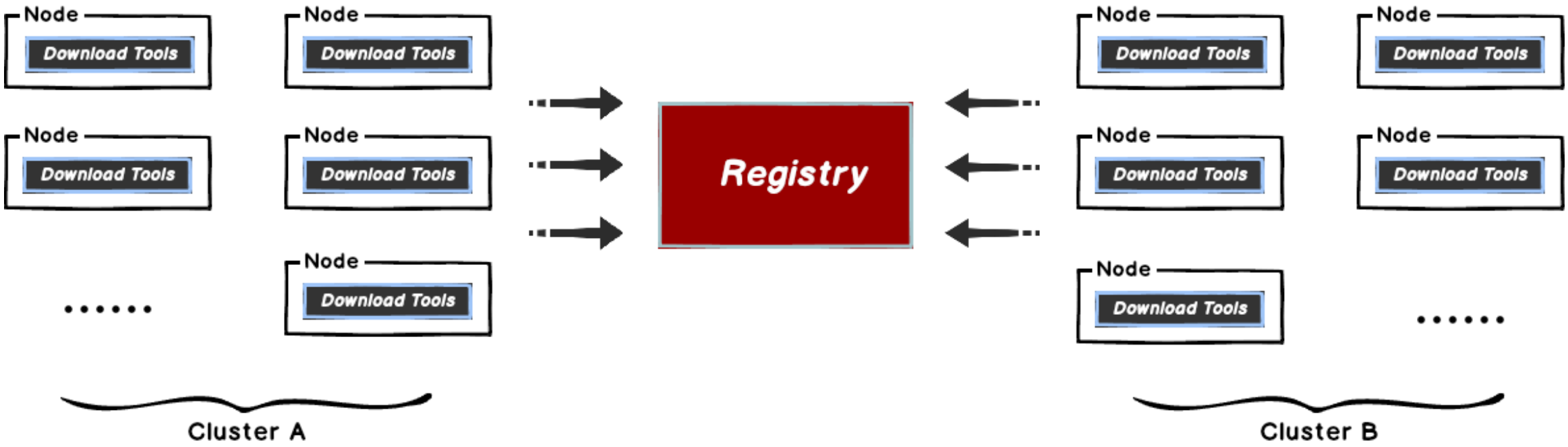
CLOUD NATIVE COMPUTING FOUNDATION

Introduction



Introduction

When downloading files in large batches, the storage bandwidth can easily reach the limit.



How to solve the problem?

Increase storage bandwidth

P2P eliminates the impact of bandwidth limitations

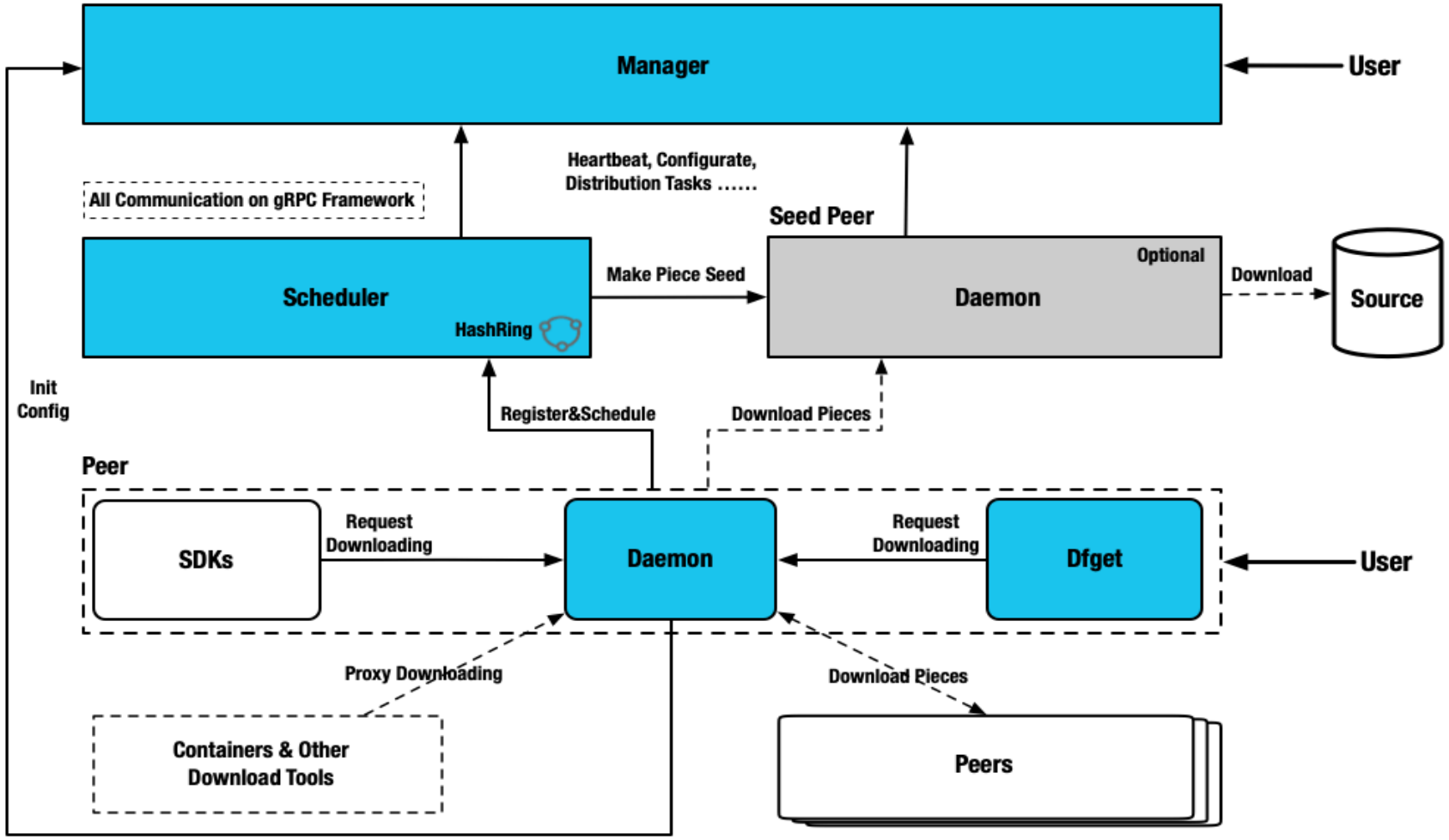
Reduce download size

Introduction

Manager manages cluster relationships, dynamic configuration and provides a console.

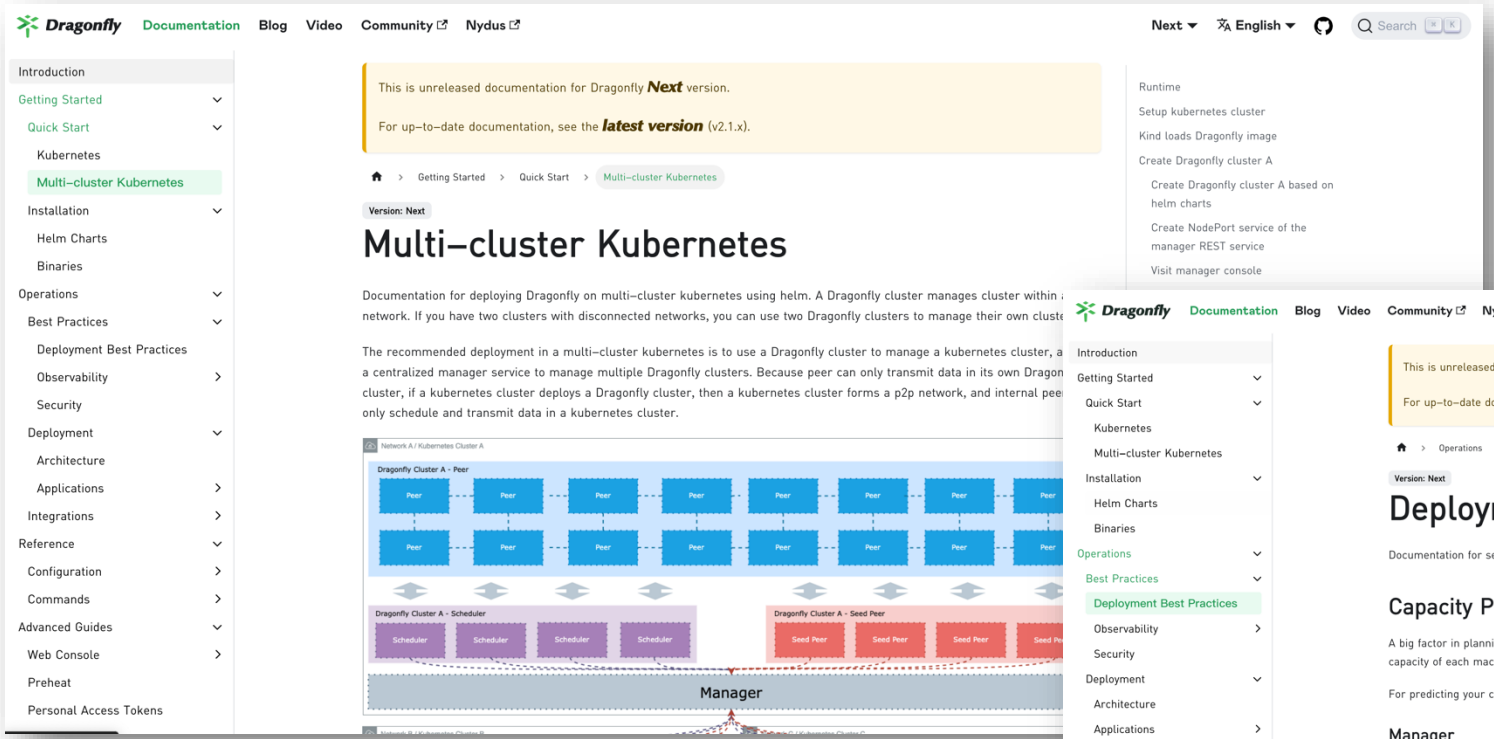
Scheduler schedules a set of parent nodes for the download node.

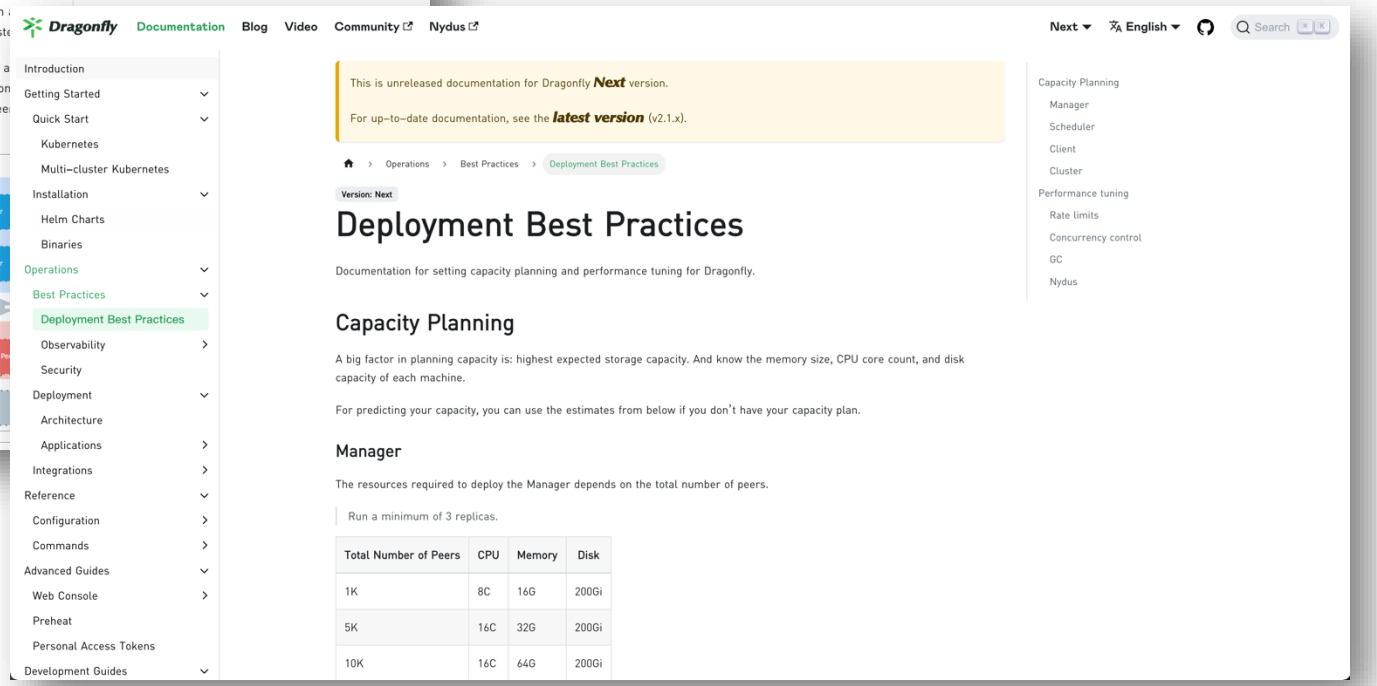
Peer provides upload and download capabilities.



Exciting Updates

Document

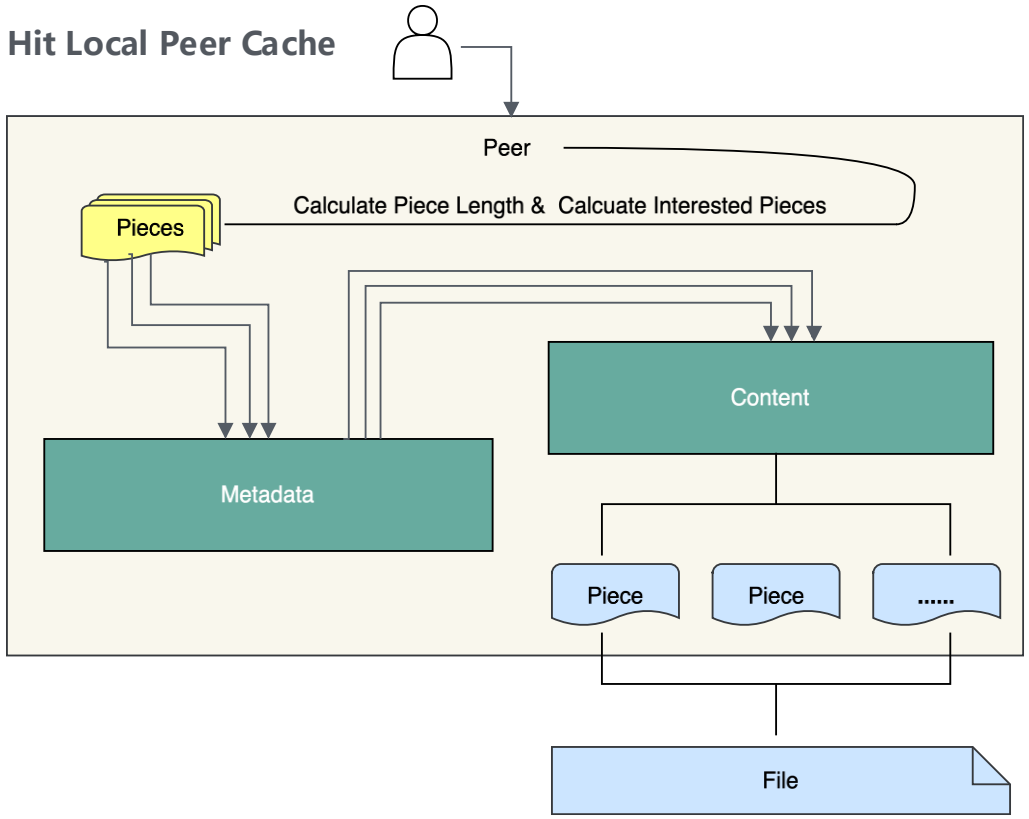




Exciting Updates

Rust Client

Hit Local Peer Cache



client Public

main | 4 Branches | 98 Tags | Go to file t | Add file | Code

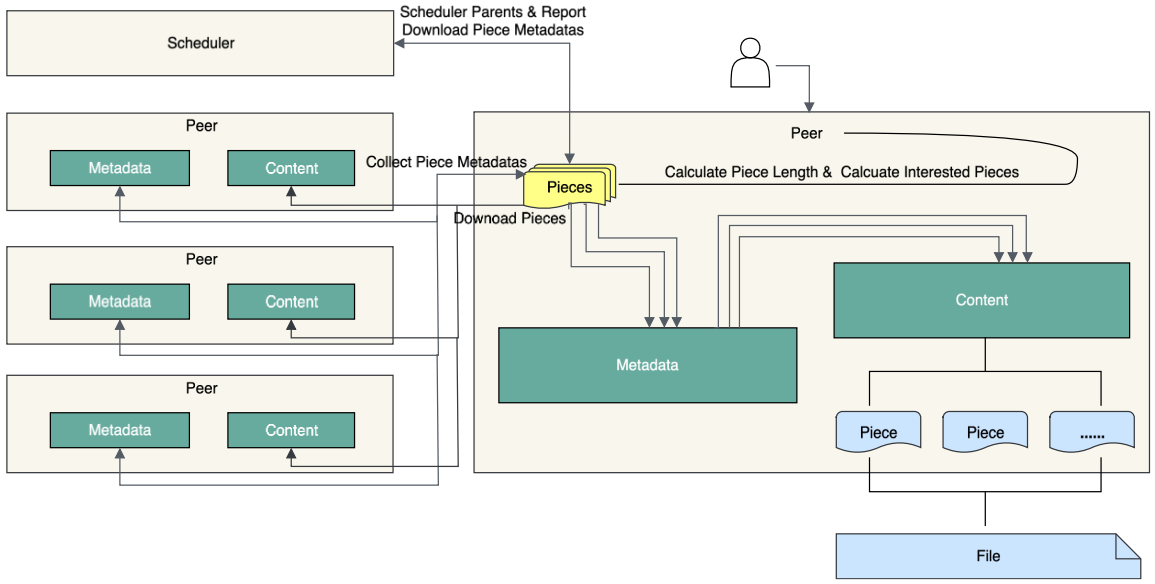
Liam-Zhao Feat: add tar package build (#659) 175be38 · 6 hours ago 591 Commits

.github	Feat: add tar package build (#659)	6 hours ago
ci	Feat: dfdaemon is started using systemd (#648)	4 days ago
dragonfly-client-backend	chore(deps): Bump libloading from 0.8.4 to 0.8.5 (#634)	last week
dragonfly-client-config	feat: when send to scheduler failed, join_set detach all ta...	19 hours ago
dragonfly-client-core	chore(deps): Bump libloading from 0.8.4 to 0.8.5 (#634)	last week
dragonfly-client-init	feat: remove header in mirror config	2 days ago
dragonfly-client-storage	feat: add log for deleting piece (#656)	3 days ago
dragonfly-client-util	feat: support crc32 for calculating piece content (#649)	4 days ago
dragonfly-client	feat: when send to scheduler failed, join_set detach all ta...	19 hours ago
.gitignore	chore: change gitignore for vendor (#415)	4 months ago
CONTRIBUTING.md	docs: fix typo in README.md (#381)	4 months ago
Cargo.lock	feat: when send to scheduler failed, join_set detach all ta...	19 hours ago
Cargo.toml	feat: when send to scheduler failed, join_set detach all ta...	19 hours ago
LICENSE	Initial commit	last year
README.md	Add license scan report and status (#611)	3 weeks ago
codecov.yml	chore: add codecov for testing (#3)	last year
rust-toolchain.toml	feat: add request timeout for connection (#627)	2 weeks ago

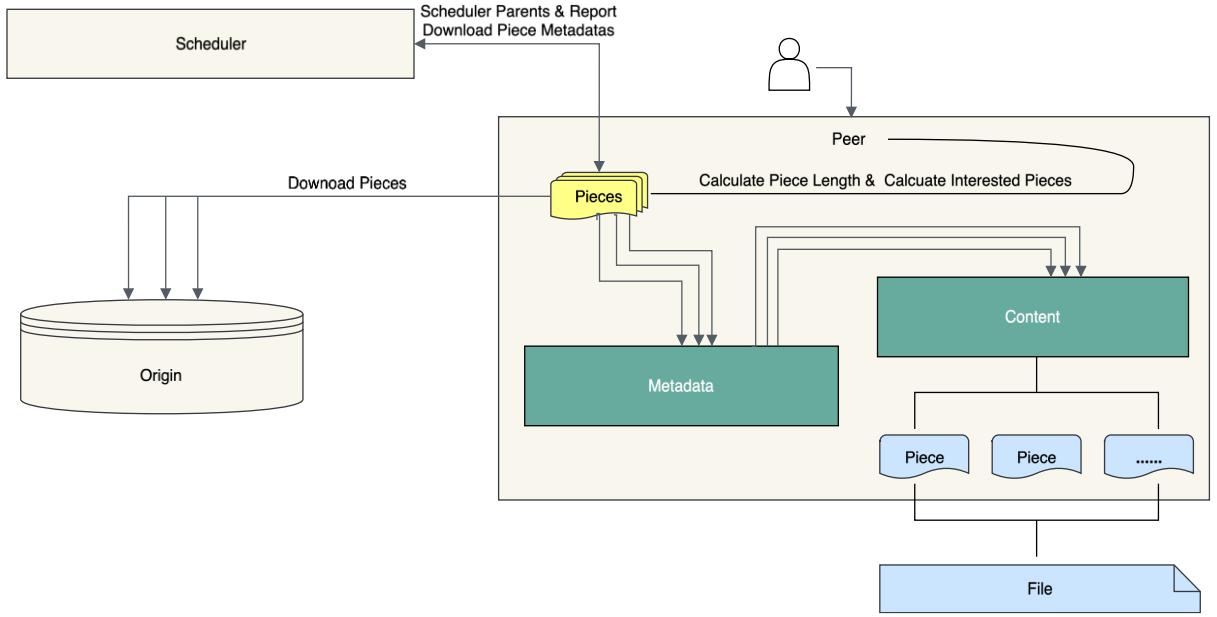
Exciting Updates

Rust Client

Hit Remote Peer Cache



Downoad From Origin



Exciting Updates

KubeCon CloudNativeCon China 2024 | OPEN SOURCE SUMMIT | AI_dev

Kubernetes 1.31: Read Only Volumes Based On OCI Artifacts (alpha)

By **Sascha Grunert** | Friday, August 16, 2024

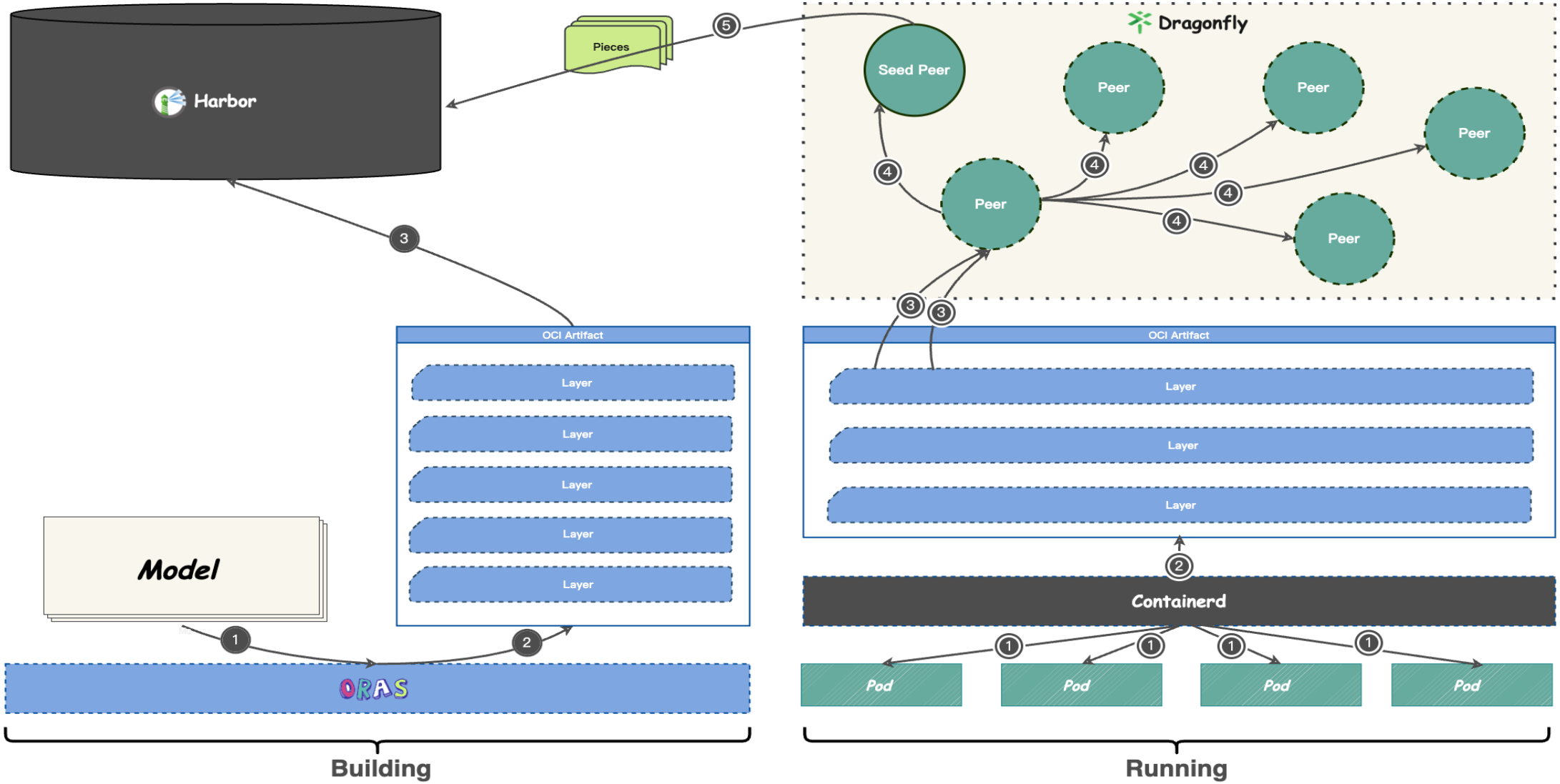
The Kubernetes community is moving towards fulfilling more Artificial Intelligence (AI) and Machine Learning (ML) use cases in the future. While the project has been designed to fulfill microservice architectures in the past, it's now time to listen to the end users and introduce features which have a stronger focus on AI/ML.

One of these requirements is to support Open Container Initiative (OCI) compatible images and artifacts (referred as OCI objects) directly as a native volume source. This allows users to focus on OCI standards as well as enables them to store and distribute any content using OCI registries. A feature like this gives the Kubernetes project a chance to grow into use cases which go beyond running particular images.

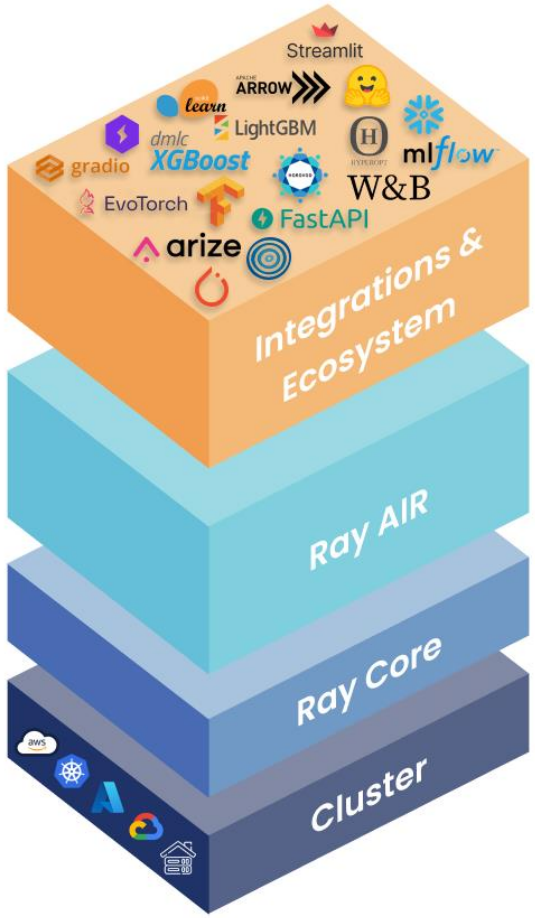
Given that, the Kubernetes community is proud to present a new alpha feature introduced in v1.31: The Image Volume Source (KEP-4639). This feature allows users to specify an image reference as volume in a pod while reusing it as volume mount within containers:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  containers:
    - name: test
      image: registry.k8s.io/e2e-test-images/echoserver:2.3
      volumeMounts:
        - name: model
          mountPath: /model
  volumes:
    - name: model
      image:
        reference: harbor.example.com/example/llama:v1.0.0
        pullPolicy: IfNotPresent
```

Exciting Updates



Introduction



a growing ecosystem of integrations and partnerships with popular projects

high-level libraries which enable simple scaling of AI workloads

a low-level distributed computing framework with a concise core, Python-first API

What is Ray?

An open source framework to build and scale your ML and python APP easily

Originated from RISELab, 32.6k github star, supporting ***100w+*** CPU core in AntGroup

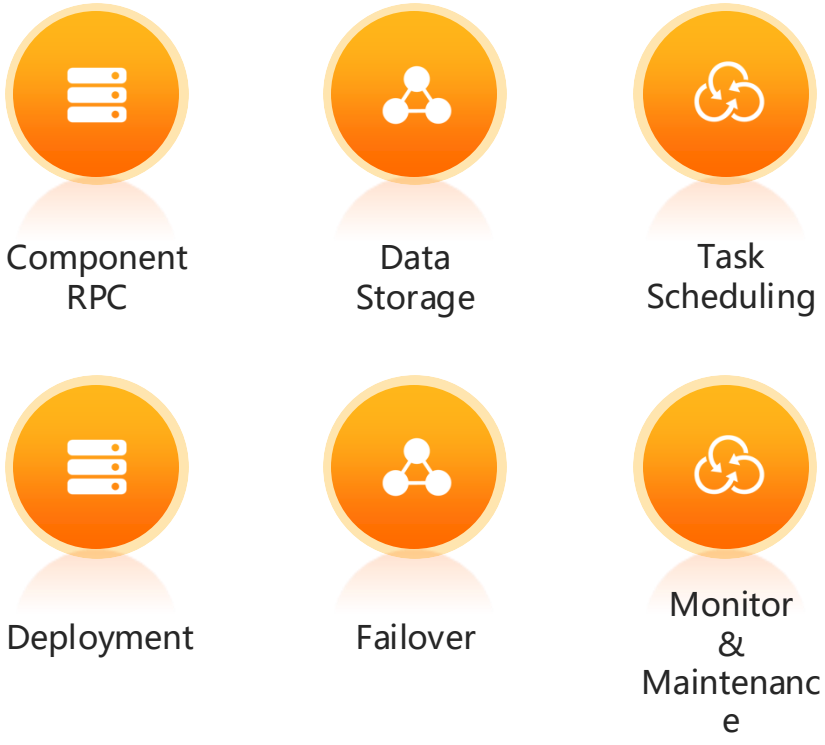
RayCore provides distributing primitives. ***RayAIR*** provides high-level library in AI workloads, e.g. traning, parameter tuning, model serving, batch processing. A rich and growing ***ecosystem*** of integrations.

OpenAI

"At OpenAI, we are tackling some of the world's most complex and demanding computational problems. Ray powers our solutions to the thorniest of these problems and allows us to iterate at scale much faster than we could before."

Greg Brockman, CTO and cofounder, OpenAI

Introduction



What capabilities a typical distributed system have?

RPC How nodes are communicated. grpc or brpc ? FlatBuffer or ProtoBuffer?

Data Storage What flows between nodes. Memory, Disk, Cloud ? Row-wise, column wise? Schema?

Task scheduling Making the distribution worthy. Yarn, K8S?

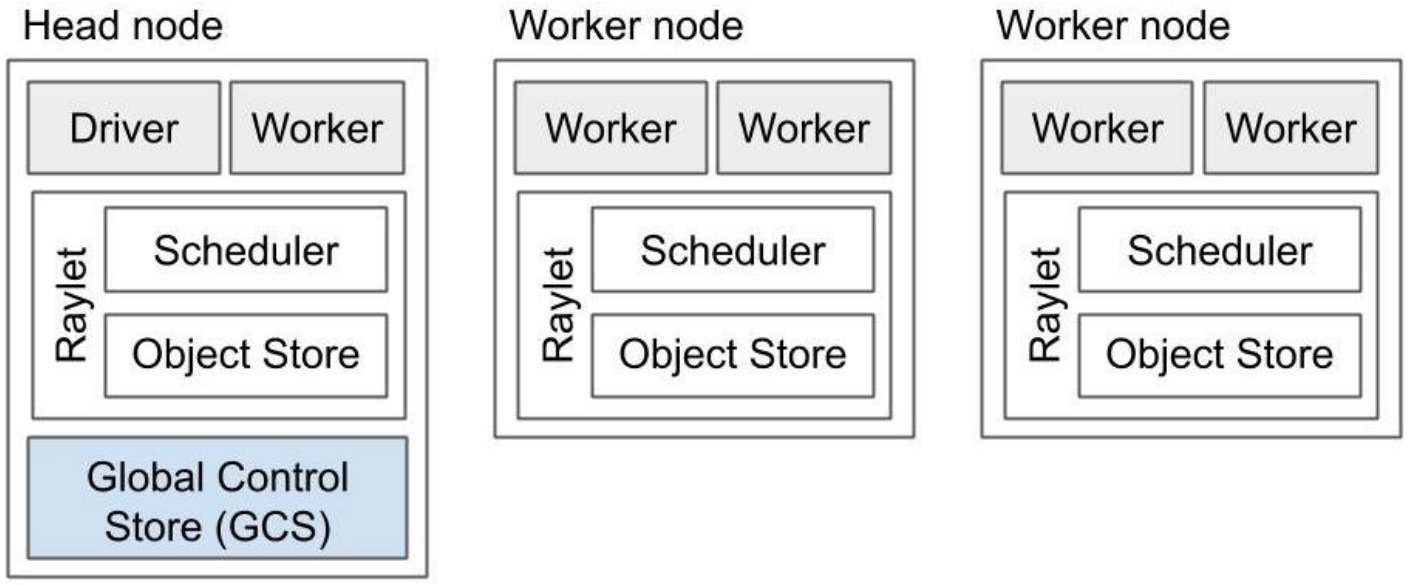
Deployment Let the system physically run

Failover The necessary part for a distributed system

Monitor Important for complex system

These are what Ray ***precisely*** provides

Architecture overview

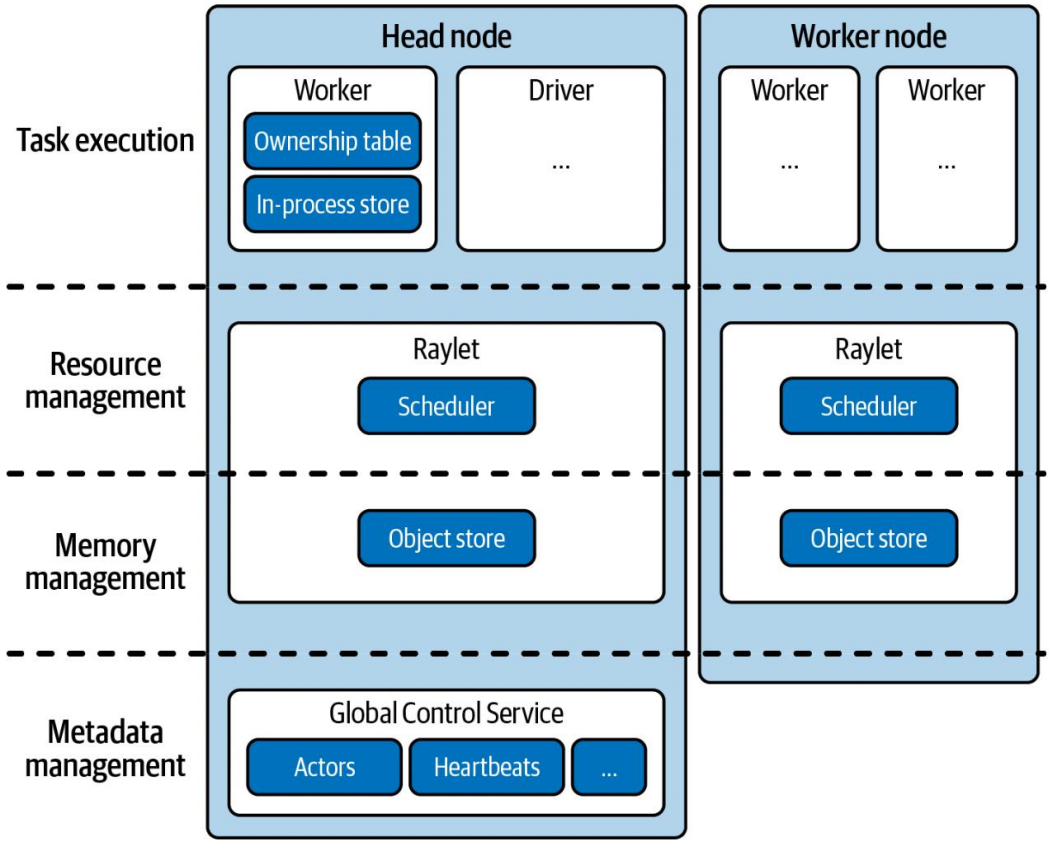


Head node : global state, top-level scheduling, job submission, resource management

Worker node : second-level scheduling, task execution.

- *Scheduler* :task scheduling & failover
- *Object Store* : data storage and transfer
- *Worker* : task execution

Ray Architecture



Head node : global state, top-level scheduling, job submission, resource management

Worker node : second-level scheduling, task execution.

- *Scheduler* :second-level scheduling
- *Object Store* : data storage and transfer
- *Worker* : task execution

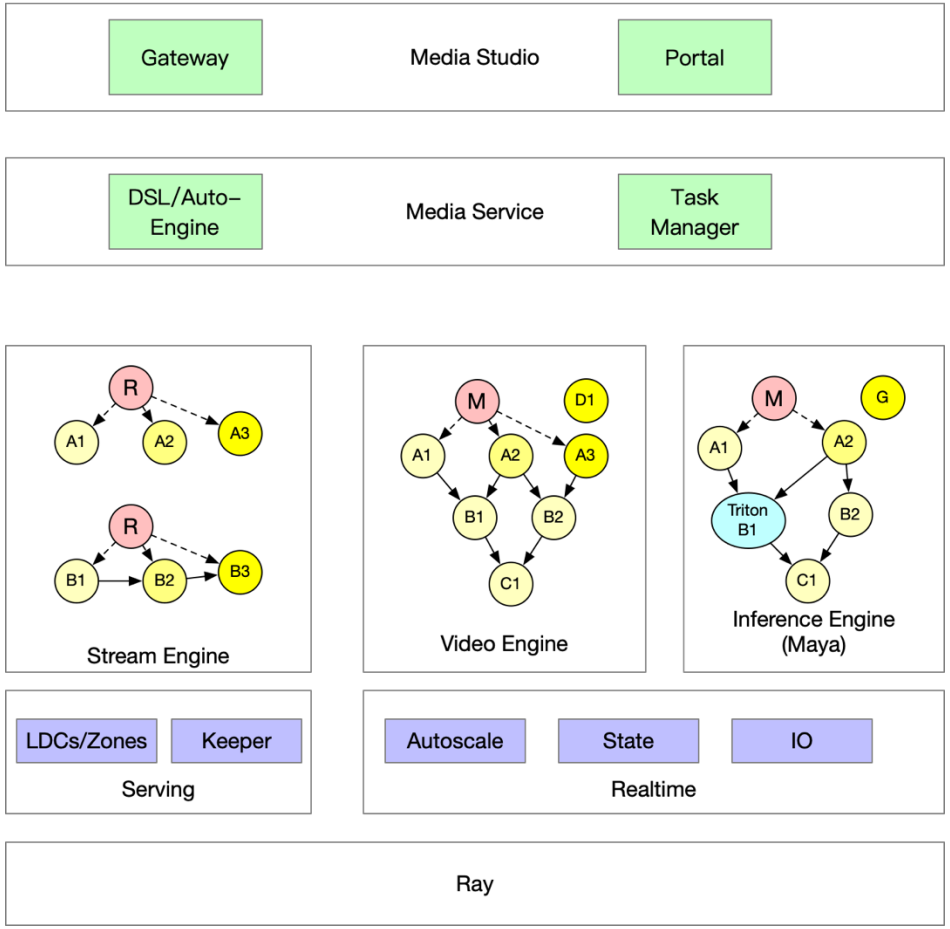
Media file processing

Stream Engine for live stream, using RayServe. *Online*

Video Engine for media file processing, e.g. IO, render, decode, AFTS. *Near online*

Inference Engine for multi-model DL, feature extraction, high-performance inference. *Offline*

The data are all disk files



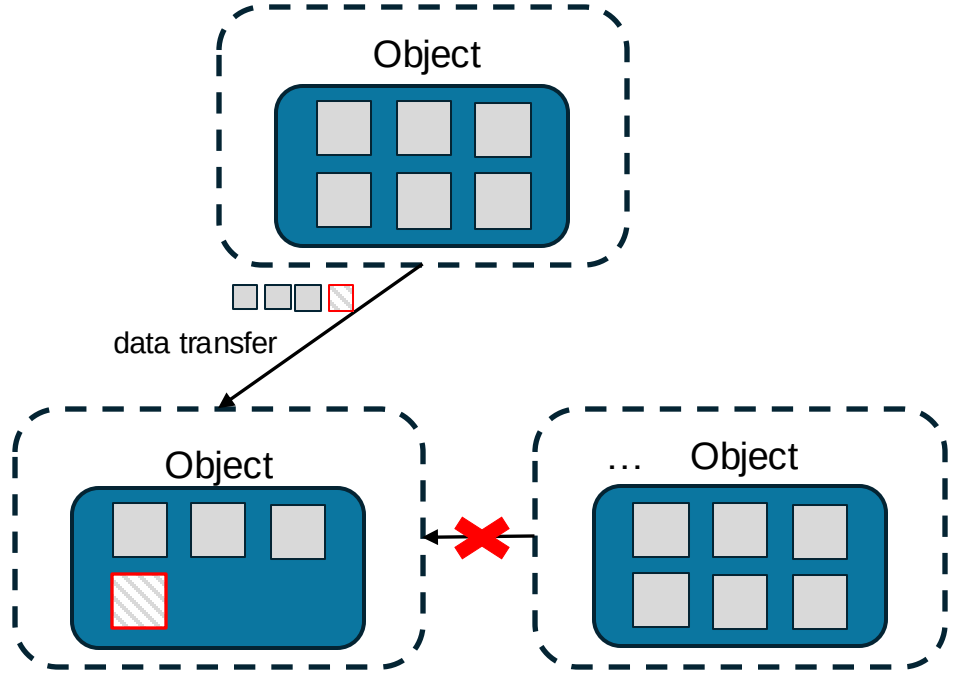
Motivation

Remote large file download Media file 1MB ~ 500MB, Model 1GB~20GB, cluster-level distribution

Directory download Model is actually a directory with model states, structure, config, etc.

File life-cycle management File is either large or frequently distributed to cluster, piling and leaking are not allowed

Natural failover The workflow may time consuming, better not start over when Pod failed



- The whole copy is transferred even though chunked
- 1-to-1 even though other nodes may have the same copy

RayFile - Ray & Dragonfly

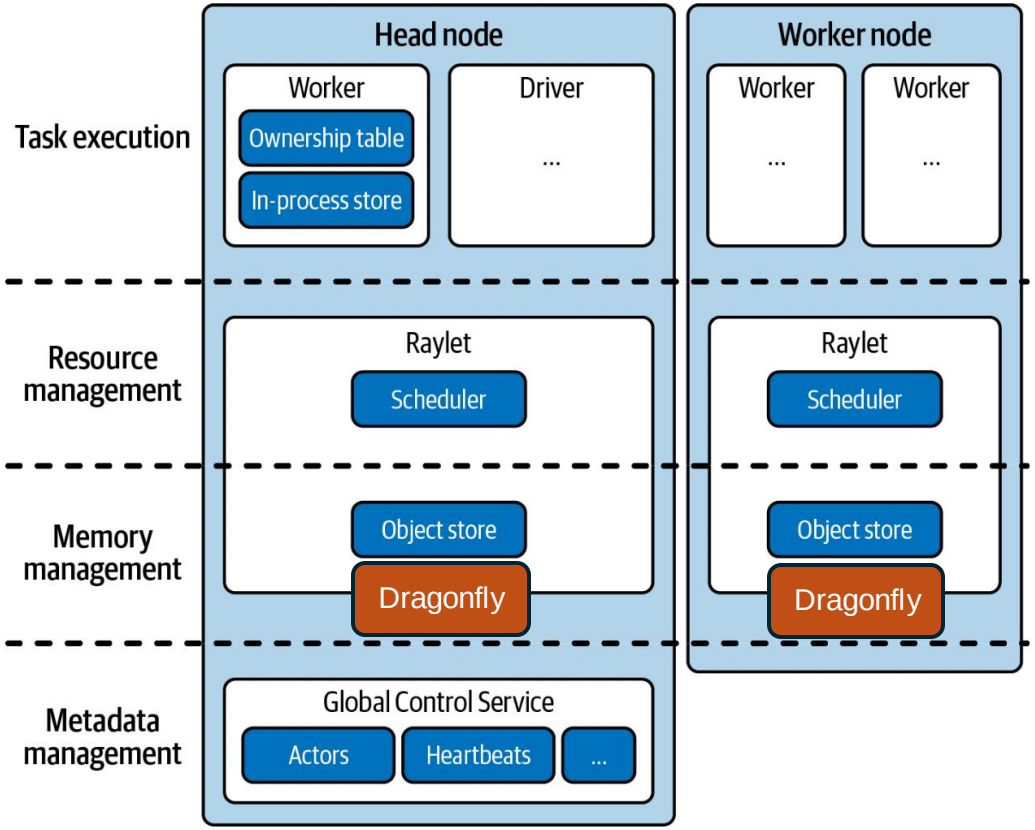
P2P transfer Data(File) chunk is distributed in multiple nodes

Native API Adapt dragonfly SDK to Ray native API

Hierarchical Cache Worker -> Job -> Ray node -> D7Y peer

GC protocol File’ s lifecycle follows the same GC protocol with Ray Object

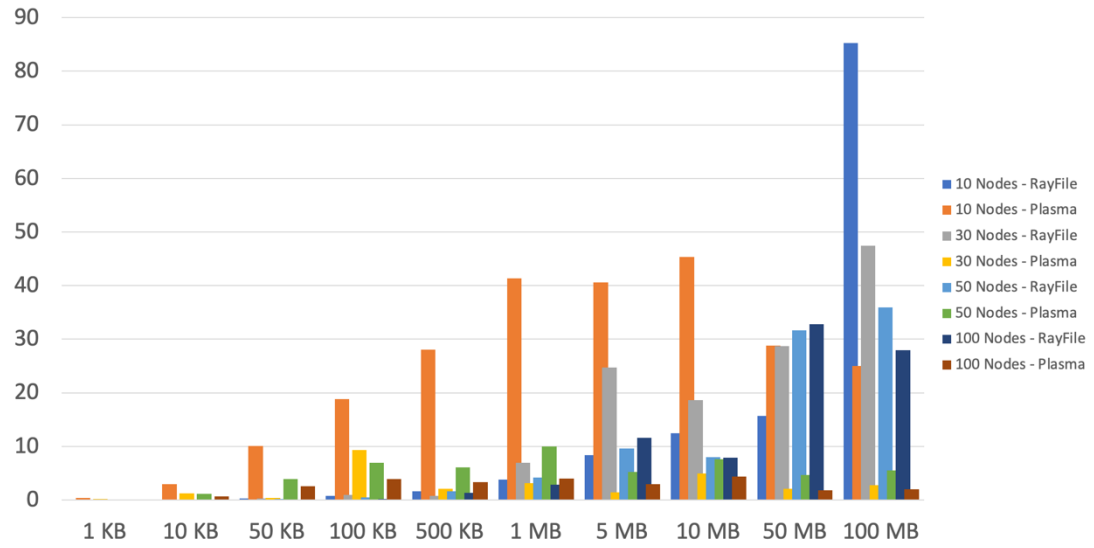
File upload local disk file, e.g. ffmpeg output file, can be uploaded to the Ray(d7y) cluster



Benchmark

Dragonfly v.s. ObjectStore

The large file size or more number of copy is, the more efficient D7Y is, max to *10x* bandwidth boost



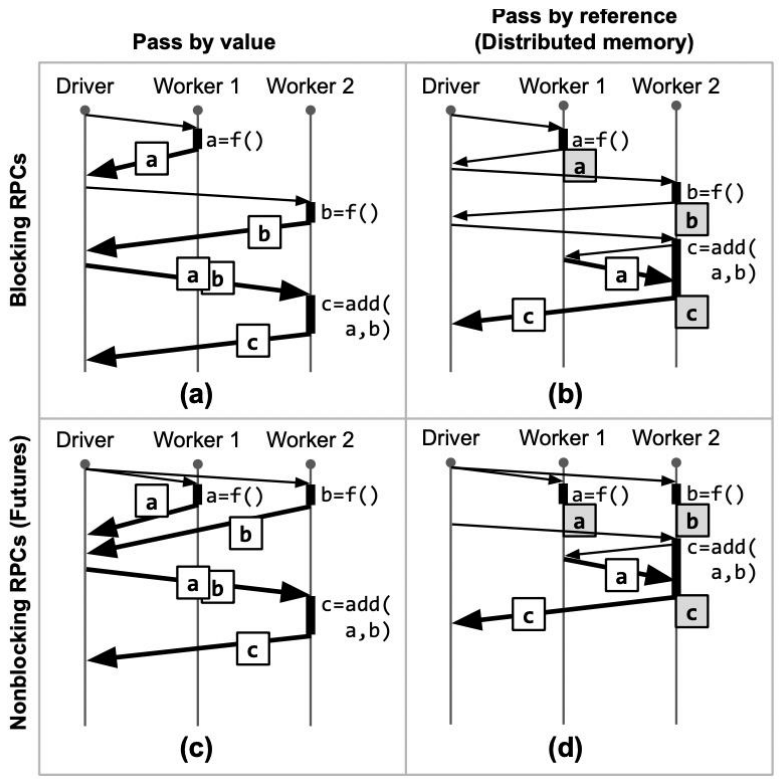
Stress test – Audio extraction scenario

File size(MB)	File amount	avg. bandwidth(MB/s)	P50 latency(s)	P90 latency(s)	P99 latency(s)
0~1	4814	35.75	0.14	1.04	30.82
16~32	2208	402.33	16.31	31.96	48.4
256-512	159	251.71	29.97	55.92	57.02
512-1024	27	114.16	49.26	50.27	50.39
1024-4096	7	170.26	49.27	49.57	49.57

Download latency is *stable* even file size grows from 512MB to 4GB ! But *unstable* when <64MB

RayFile/Dragonfly is incline to *large file* from both bandwidth and lantency perspective.

Remote Distributed Future



In Ray, the data that flows between tasks are just the *future reference* whose value are resolved in daemon. A task will only be executed by the time its input data’ s future is resolved

RayFile can potentially implement the mechanism since d7y is also transferring data in daemon and provides future reference, e.g. *cid*


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
China 2024
Thanks!
Dragonfly