KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

# Revolutionizing Scientific Simulations with Argo Workflows

Shuangkun Tian & Yashi Su
Alibaba Cloud

# About Us


Shuangkun, Tian
Alibaba Cloud Software Engineer
Argo Community Maintainer

Yashi, Su
Alibaba Cloud Software Engineer
Argo Community Contributer

# Agenda



1. Characteristics of Scientific Simulations
2. Why Choose Argo Workflow?
3. Challenges, Reflections, and Best Practices
4. Demo:  Molecular Dynamics Simulation

# Agenda



1. **Characteristics of Scientific Simulations**
2. Why Choose Argo Workflow?
3. Challenges, Reflections, and Best Practices
4. Demo:  Molecular Dynamics Simulation

# 1. What is Scientific Simulations




Fig 1. An Example of A Scientific Simulation

# Agenda



1. Characteristics of Scientific Simulations
2. **Why Choose Argo Workflow?**
3. Challenges, Reflections, and Best Practices
4. Demo:  Molecular Dynamics Simulation

# 2. Container-Native Workflow Engine



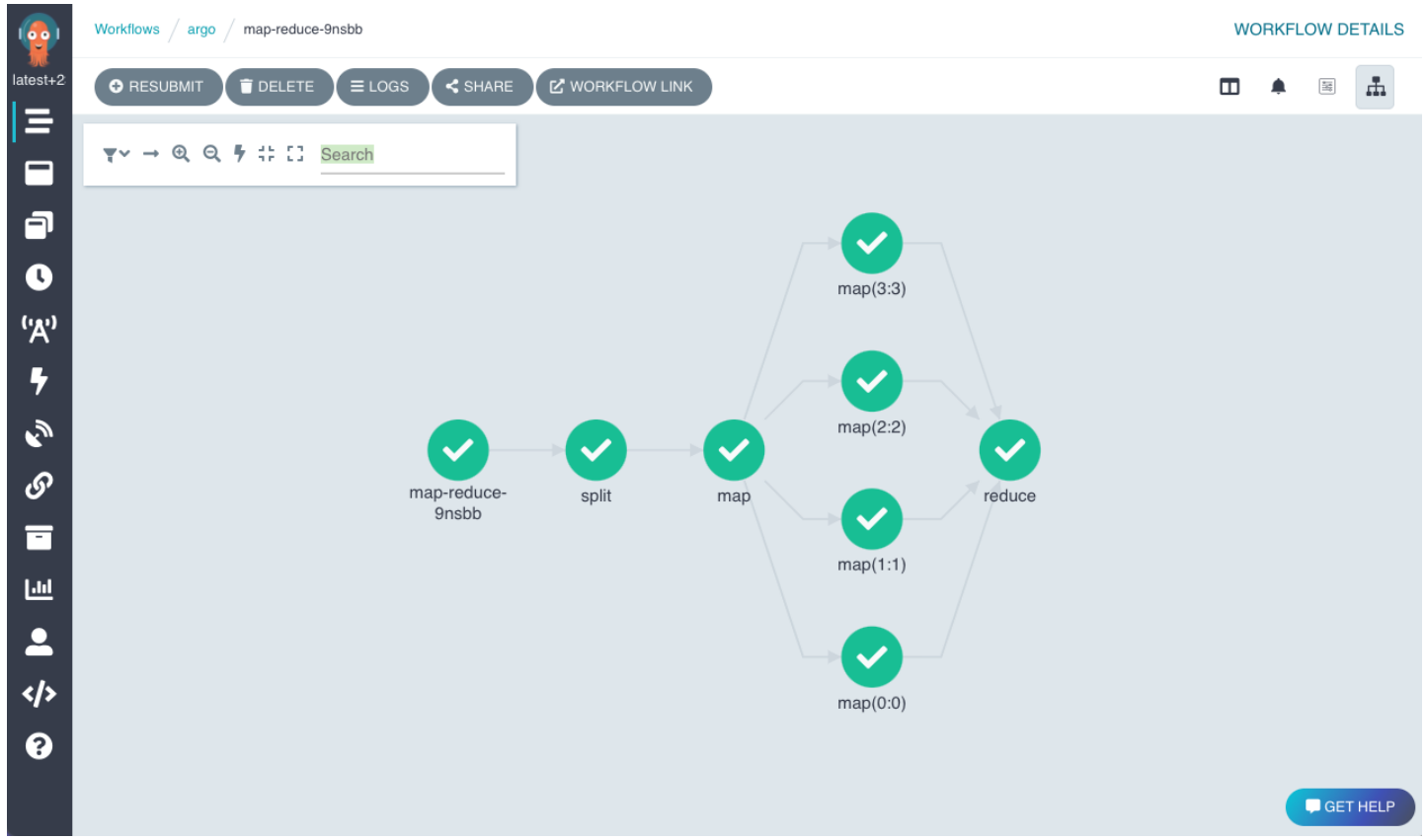



Fig 1. UI Interface of An Argo Workflow

argo

**Use Cases:**

- Machine Learning Pipelines
- Data / Batch Processing
- Infrastructure AutoMation
- CI / CD

# 2. Workflow Definition



```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-
spec:
  entrypoint: hello-world-example
  templates:
  - name: hello-world-example
    steps:
    - - name: generate-artifact
        template: whalesay
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["cowsay hello world | tee /tmp/hello_world.txt"]
    outputs:
      artifacts:
      - name: hello-art
        path: /tmp/hello_world.txt
status:
  nodes: hello-world-example-6ftnv
  taskResultsCompletionStatus: hello-world-example-6ftnv: false
  phase: Running
```

**Depenency Definition:**
Serial dependencites / Complex DAGs

**Template:**
Image, Arguments,
Inputs, and Output

**Status:**
Nodes, Phase and
TaskResultsComletionStatus

# 2. Why Argo Workflows



- Kubernetes-Native
- One of the Most Active Communities
- Scaling for High Elasticity
- Support CI / CD
- Rich Ecosystem

Argo Events

# Agenda



1. Characteristics of Scientific Simulations
2. Why Choose Argo Workflow?
3. **Challenges, Reflections, and Best Practices**
4. Demo:  Molecular Dynamics Simulation

# 3.1 Challenges of Complexity

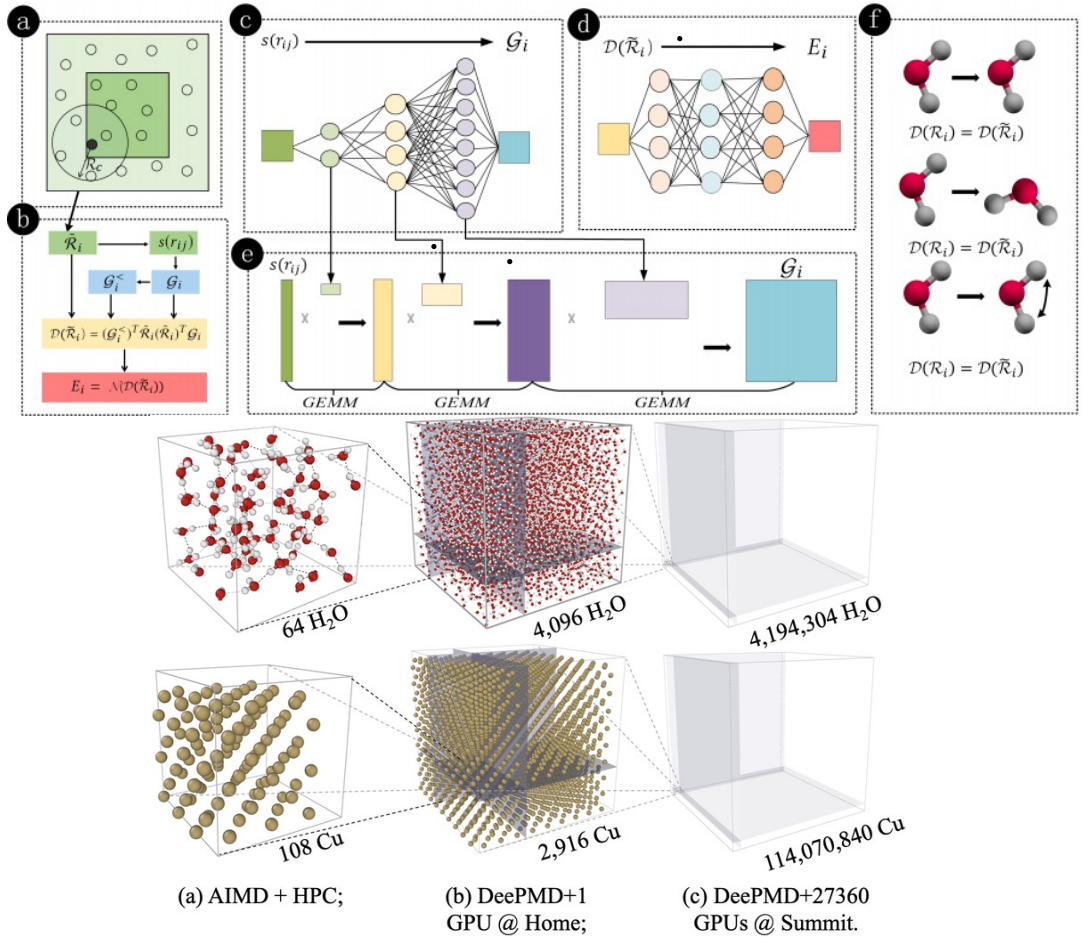



Fig 1. Typical Scientific Simulations

- **Highly Complex System**

  More than 1000 Tasks in One Step

- **Large time expenditure**
- **Massive Parameters**

  Scientific Simulations Require A Great Amounts of Arguments


Guo Z, Lu D, Yan Y, et al. Extending the limit of molecular dynamics with ab initio accuracy to 10 billion atoms[C]//Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2022: 205-218.

Lu D, Wang H, Chen M, et al. 86 PFLOPS deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy[J]. Computer Physics Communications, 2021, 259: 107624.

# 3.1 Support Large-Scale Workflow

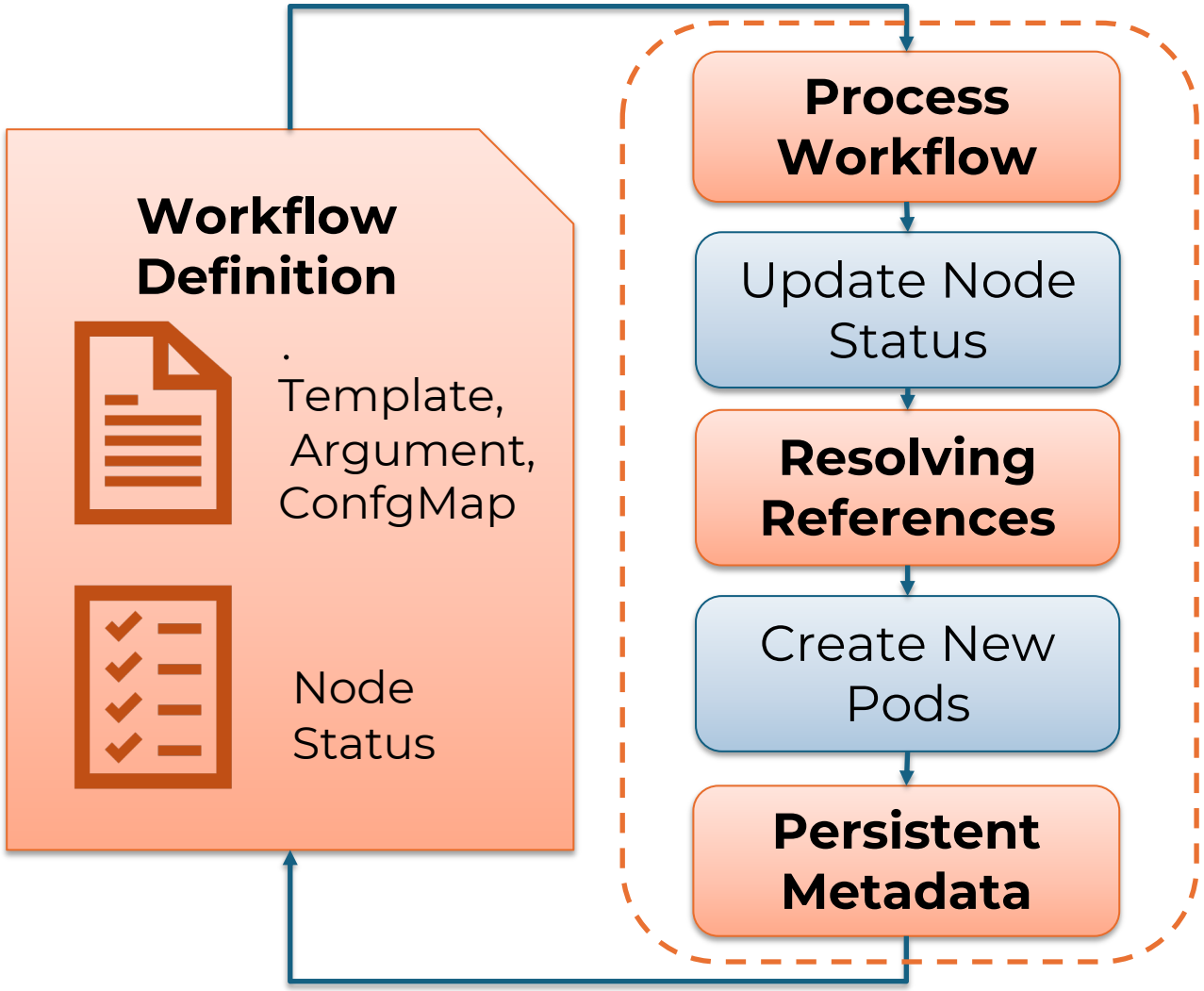



Fig 1. Large-Scale Workflow Processing Diagram

- **Input Arguments Length Limit**

=> Offload arguments to ConfigMap.

.

- **Resolving References Timeout**

=> Changing to parallel operation.

- **Metadata Explosion**

=> Offload metadata to database.

# 3.1 Improvements

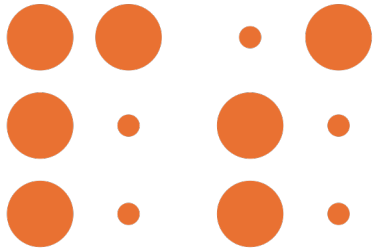


**Expanding Arguments**

The startup arguments can reach 1MB

**Accelerated Startup**

5,000 Pods can be started within 1 minute



**Amount of Pods**

Increased from 10,000 to 40,000 within a Workflow

# 3.2 Challenges of Exceptions



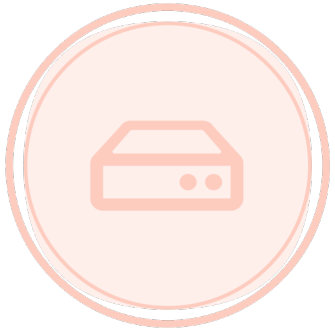
## Exceptions that May Encountered ...

**Service Interruption**
Server Restart, Network Jitter

**System Compatibility**
Images do NOT Match the OS




**Inventory Shortage**
Required Instances are Out of Stock



**Resource Exceeded**
Incorrect Estimation of Required Resources

# 3.2 Different Retry Scenarios



- Workflow consists of lots of steps.
- Easy to Retry Successful:
    1) Service Interruption
    2) Network Jitter
- Require Manual Intervention:
    1) Resource Exceeded
    2) Out of Stock

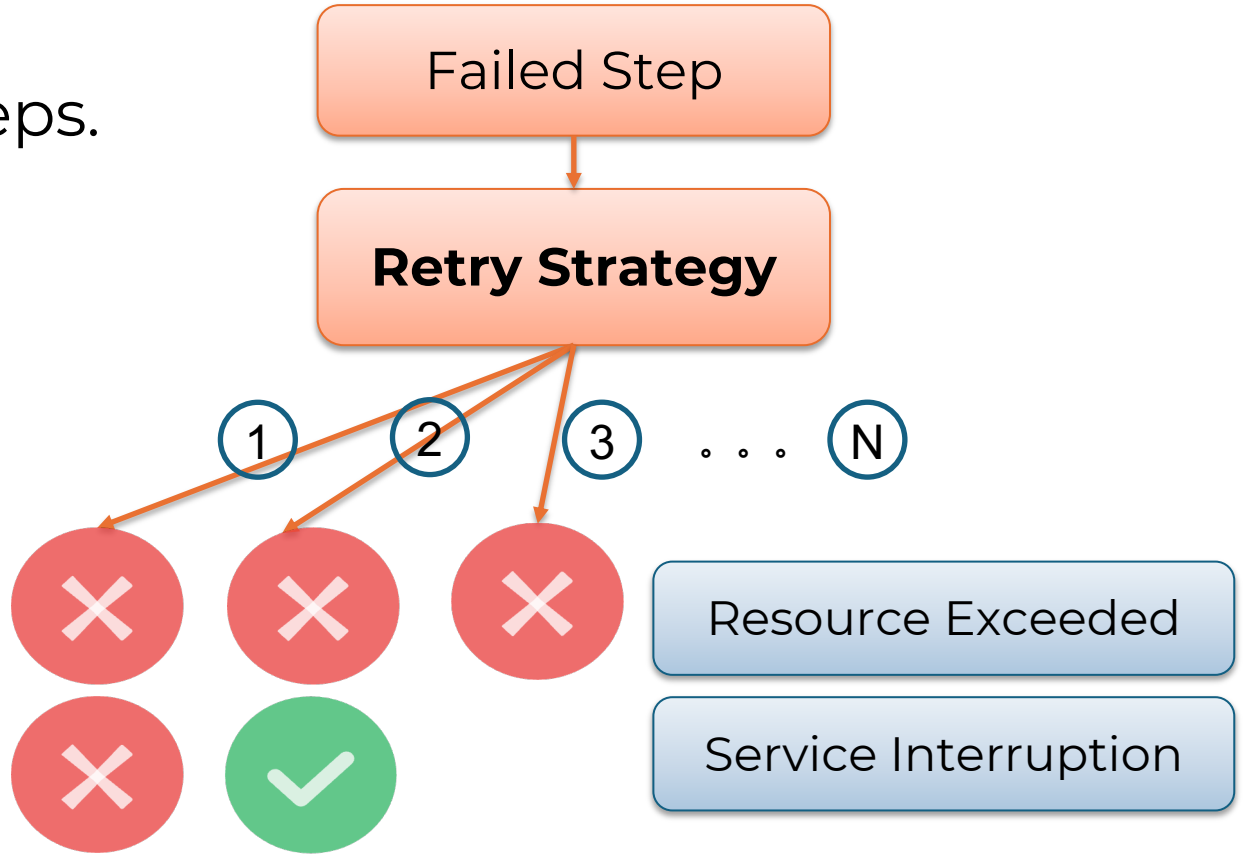



Fig 1. Retry Scenarios With Different Success Rates

# 3.2 Reasonable Retry



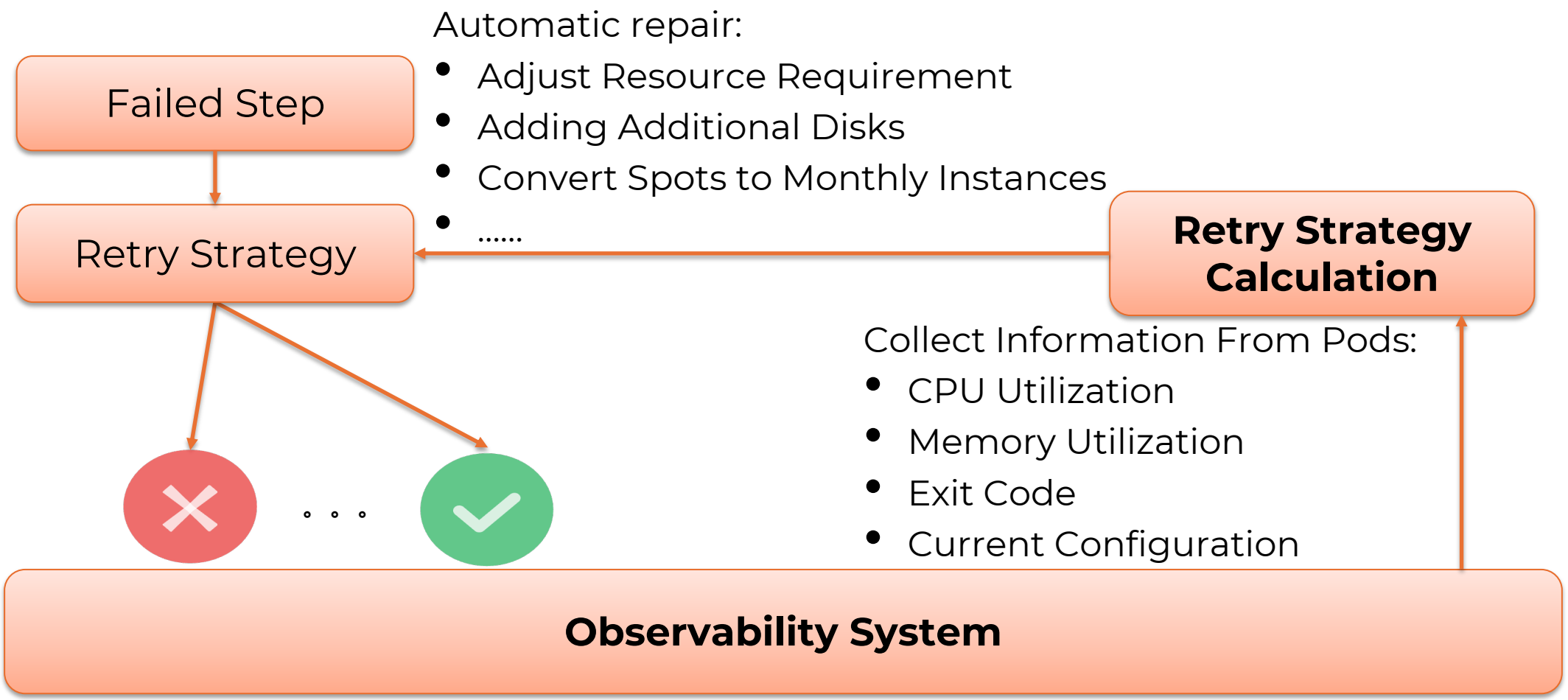



Fig 1. Retry Scenarios With Different Success Rates

# 3.3 Large-Scale Data Interaction

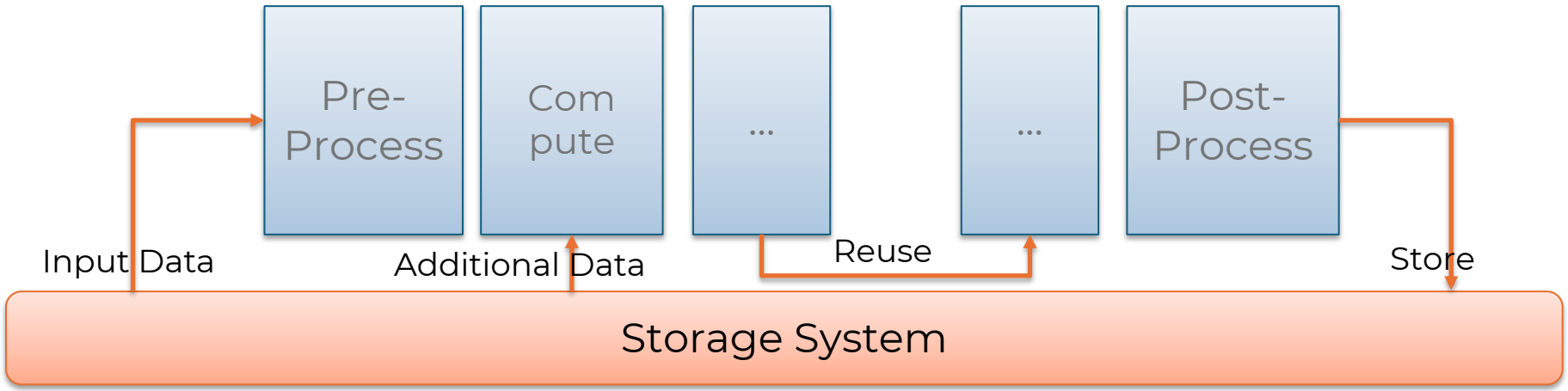



Fig 1. Large-Scale Data Interaction

**Storage System in Workflow:**

- Should be Shareable, Easy to Use
- Acceptable Storage Cost
- Will Greatly Affects Performance

# 3.3 Data Exchange

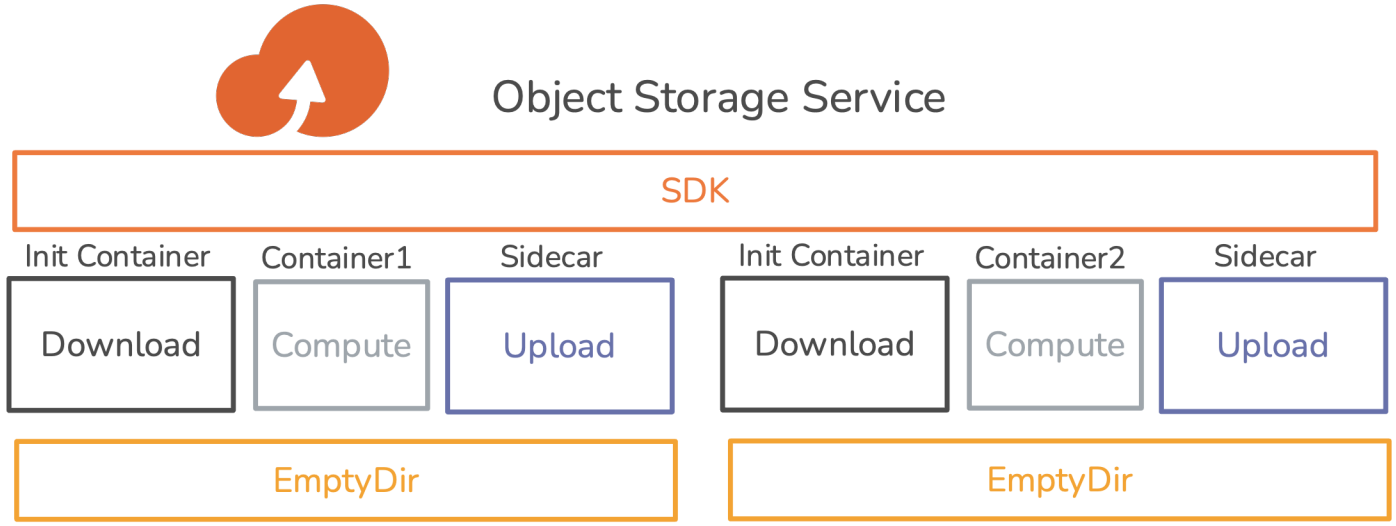



Fig 1. Data Exchange with Argoexec

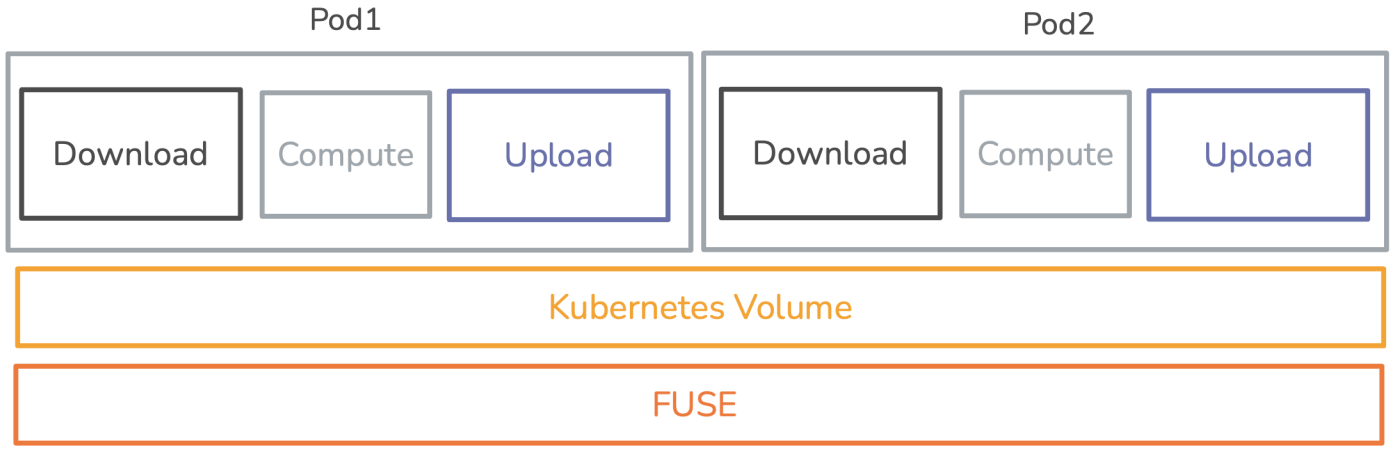



Fig 2 Data Exchange with Read-Write Volume

Argoexec :

- Independent container
- Data Processing is not shared between Templates
- May result in **duplicate data fetching**

Volume :

- Lazy data fetching
- FUSE: trade-off of **POSIX compatibility & performance**
- Shared intermediate data and **Cache**

# 3.3 Read-Write Separation

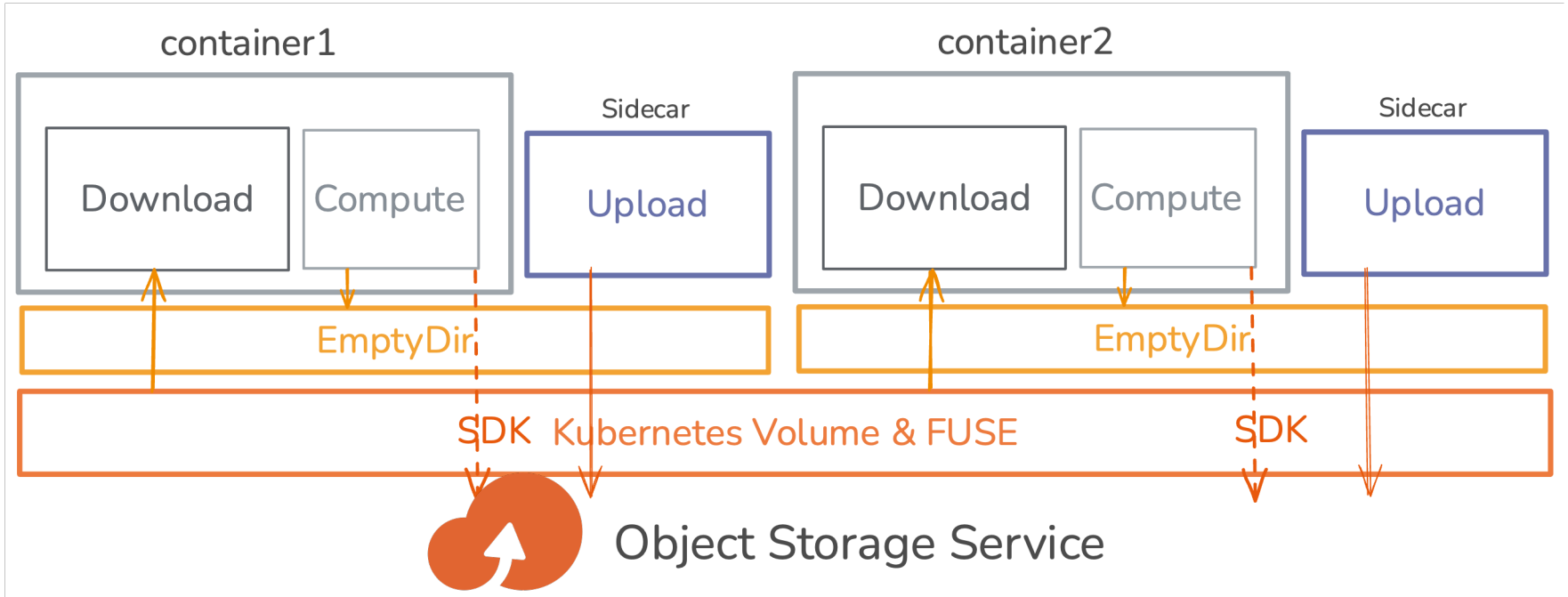



Fig 1. Read-Write Separation

- Download input files via volume in **read-only** mode.
- Upload output files or updated data via Argoexec.
- For valuable intermediate outputs, use SDK or a writable volume for upload.

# 3.3 Optimization for Read-Only Volume

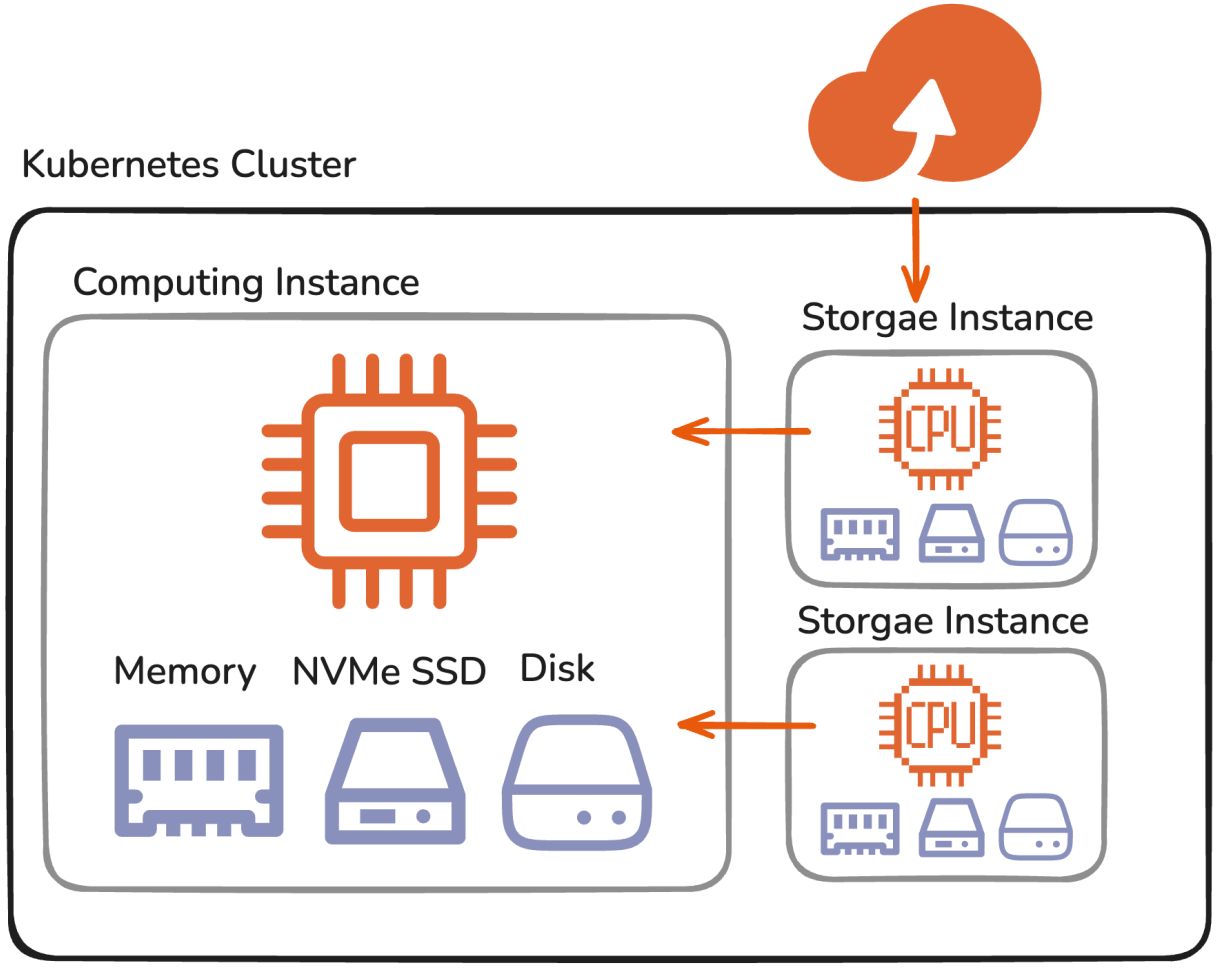
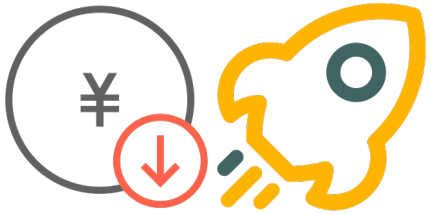



Fig 1. Cache System in a Kubernetes Cluster

- More lightweight FUSE mount tools.
- **Kernel Cache** implemented by FUSE
- **SSD, Disk Cache** by mount tools
- **Distributed Cache** within the cluster:
  - Cold Data :
    - Offload to cache workers
    - Concurrent data downloading
  - Cached Data:
    - Shared among computing workers
    - In-cluster data communication




**Lower Cost**

**Higer Performance**

# 3.3 Enhanced Memorization



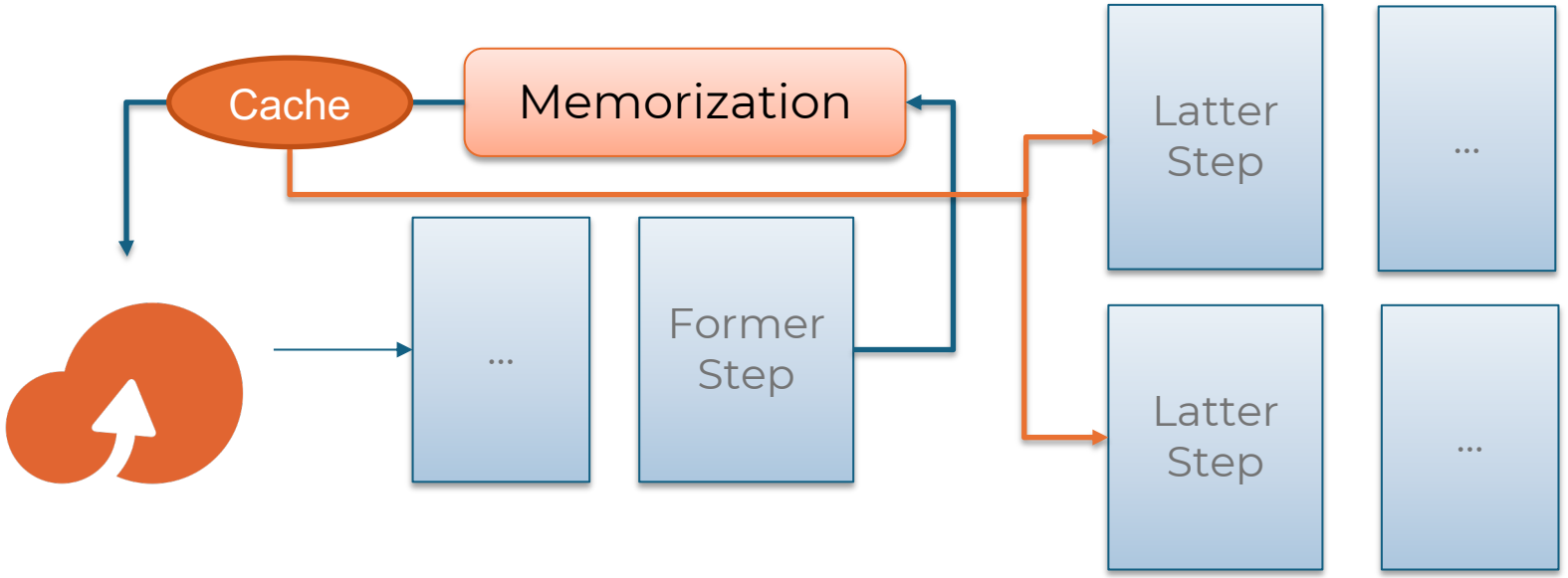



Fig 1. Read Middle Results from Cache with Enhanced Memorization

- Memorization: record metadata for reuse.
- Enhancement:
  - Namespaced, keep consistent with PVC.
  - cache data in local or in-cluster storage device.
- **Best-Effort:** clean no longer hot cache and degenerate into normal Memorization.
- **Guaranteed**: clean cache when Memorization is expired.

# 3.4 Ease of Use for Researchers

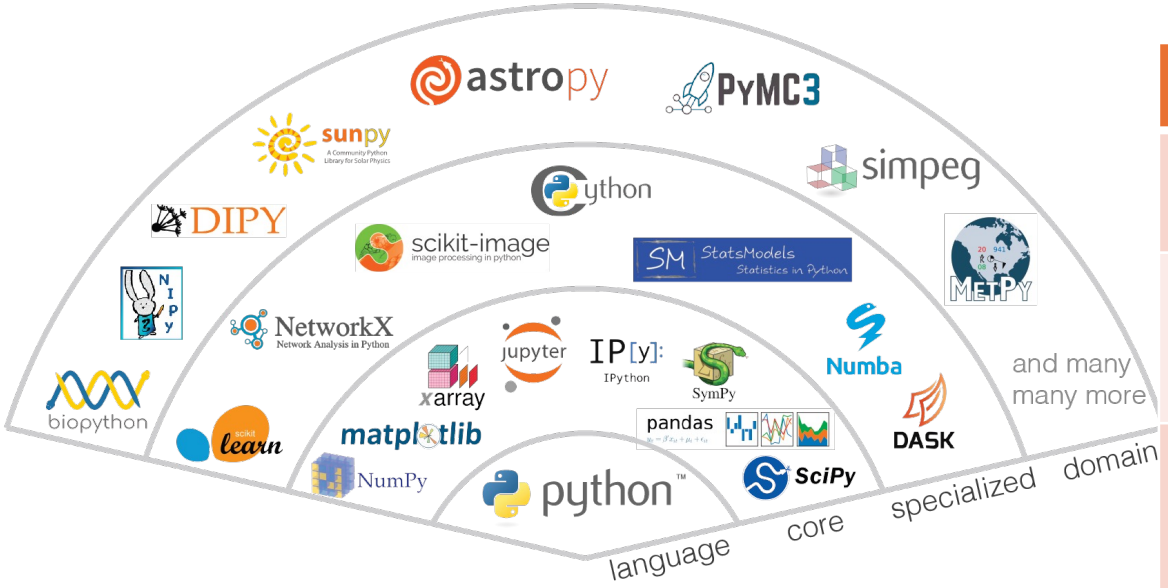



Fig 1. Python Scientific Ecosystem

tab 1. Python SDK vs YAML

| | YAML | Python SDK |
|---|---|---|
| **Conciseness** | High | High, less code |
| **Difficulty in Complex Workflows** | Hard | Easy |
| **Difficulty in Ecosystem Integration** | Hard | Easy, with rich Libraries |
| **Testability** | Hard, with frequent grammatical errors | Easy |

https://jupytearth.org/jupyter-resources/introduction/ecosystem.html

# Agenda



1. Characteristics of Scientific Simulations
2. Why Choose Argo Workflow?
3. Challenges, Reflections, and Best Practices
4. **Demo: Molecular Dynamics Simulation**

# 4 Demo: Overall Framework



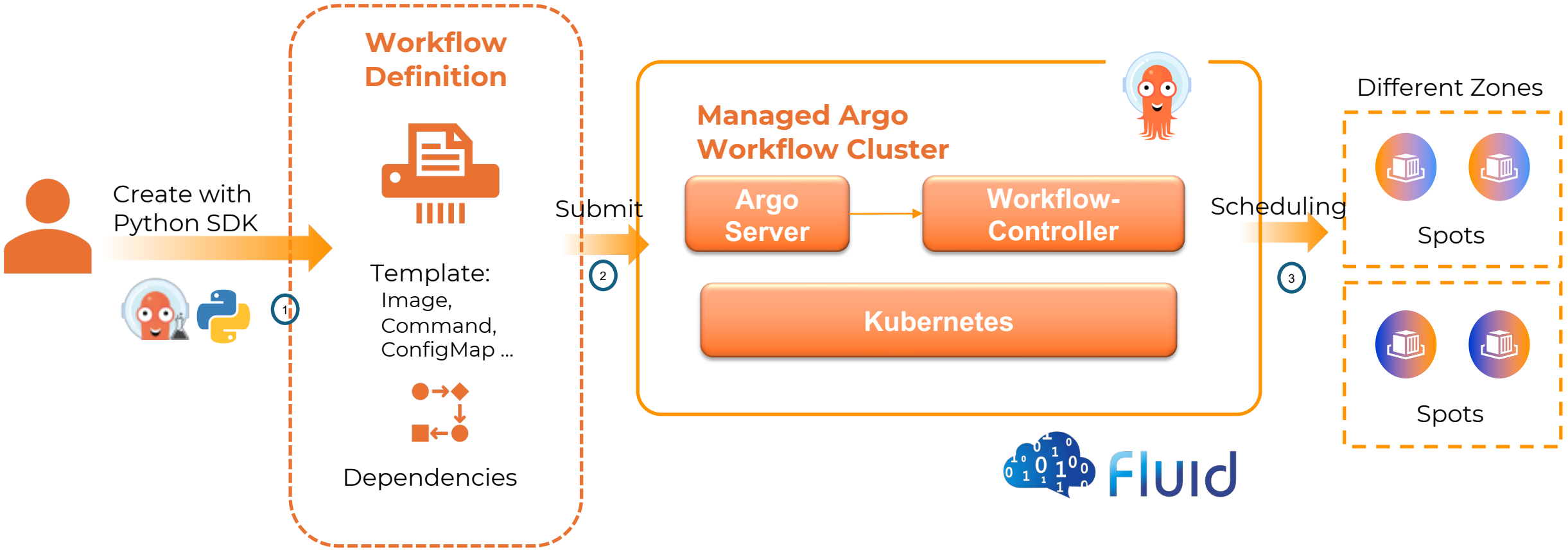
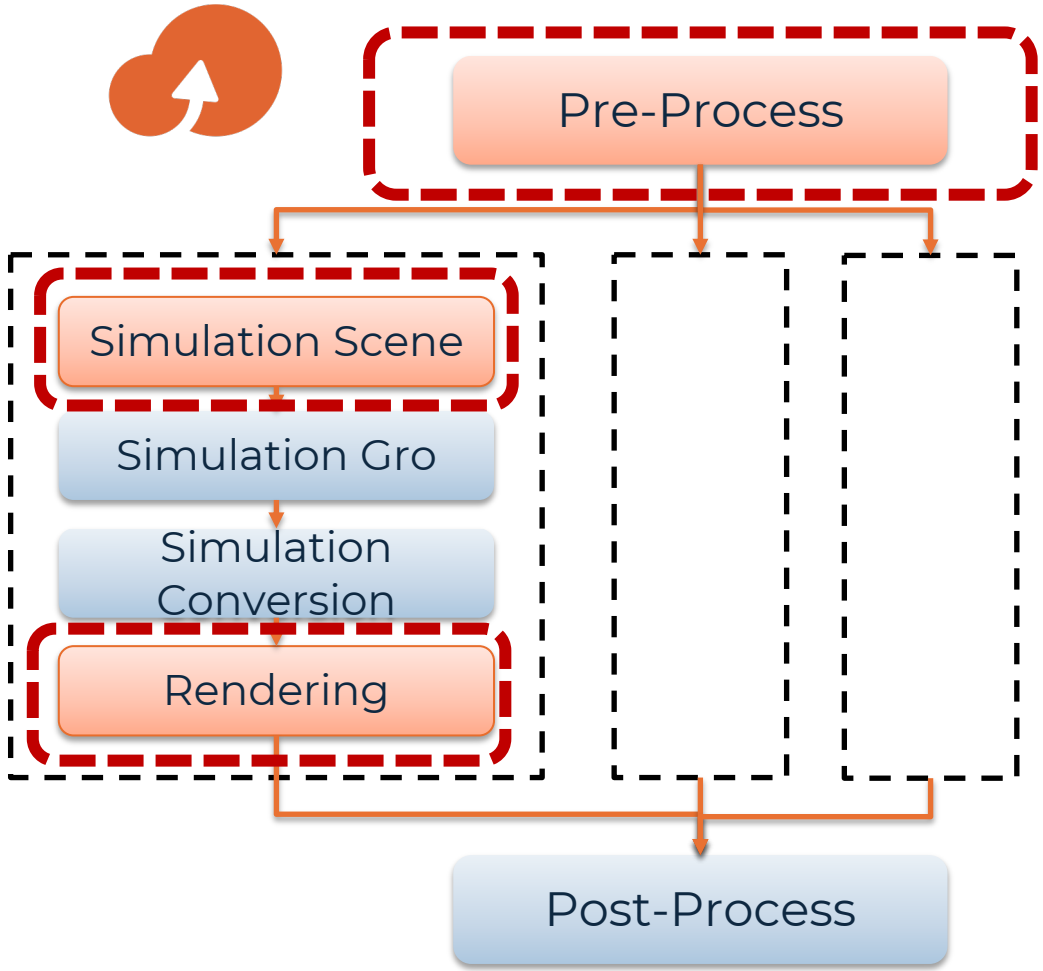



Fig 1. Overall Framework

# 4 Demo: Molecular Dynamics Simulation






1. Around 11 Gi input data.
2. Data preprocessing and classification.
3. Each process includes 3 simulations
4. Rendering and combination.

Use **Distributed Cache** to speed up **Pre-Process.**

Add **Memorization** to record template of **Simulation Scene.**

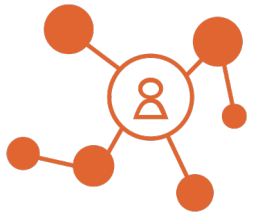
Cause OOM with unproper memory limit for **Rendering** leading to **Retry**.

Fig 1. A Molecular Dynamics Simulation for Liquid Water

# Summary



- Record Offload & Parallel Resolving References
  => Support Large-Scale and Complex Tasks



- Reasonable Retry
  => Improve Success Rate and Stability

- Best Practices for Data Stream Acceleration
  => Lower Cost, Higher Performance

- Python SDK
  => Ease to Use, Embrace Python Ecosystem


KubeCon
CloudNativeCon
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
AI_dev
Open Source GenAI & ML Summit
China 2024
Thanks !