TEADAL

Optimizing Data Pipelines in Federated and Computing Continuum Settings

Josep Sampé, Ronen Kat, Bruno Wassermann - IBM Research

Milan, March 2024

WWW.TEADAL.EU

Multi-location data pipelines

Study and propose capabilities to run data pipelines on the stretched (multi-location) data lake

What is a Multi-location data pipeline?

- A data integration flow (e.g., ETL, ELT etc.) that spans **multiple clouds or sites in a stretched data lake**
- Large-volume, geographically distributed data sources and targets
- Focus is on structured data
- Presents opportunities to partition data pipeline to address data gravity and friction

Multi-location data pipelines - Challenges

- How do I (pipeline designer / operator) decide where to run each task?
- Is my deployment compliant with privacy requirements?
- Is there a more (resource, cost, time, etc.) efficient deployment?
- How do I actually deploy and execute pipelines across locations?

Multi-location data pipelines - Solution

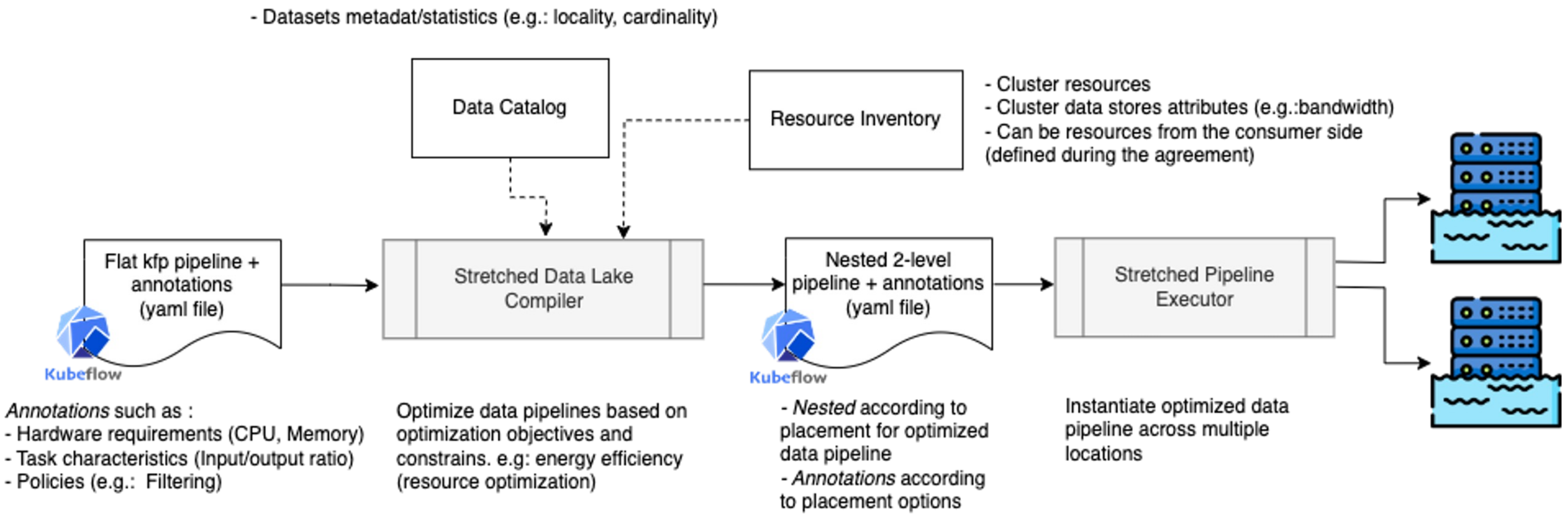
Given:

- Data pipeline specification
- Inventory of Teadal Nodes (sites)
- Constraints (e.g., compliance)
- Policy driven transformation (e.g, remove PII)
- Optimization target: data transfer volume, cost, execution time

Propose, deploy, and execute:

- Deployment plan of data pipeline tasks across sites meeting constraints and optimization target as optimally as possible

Pipeline Architecture

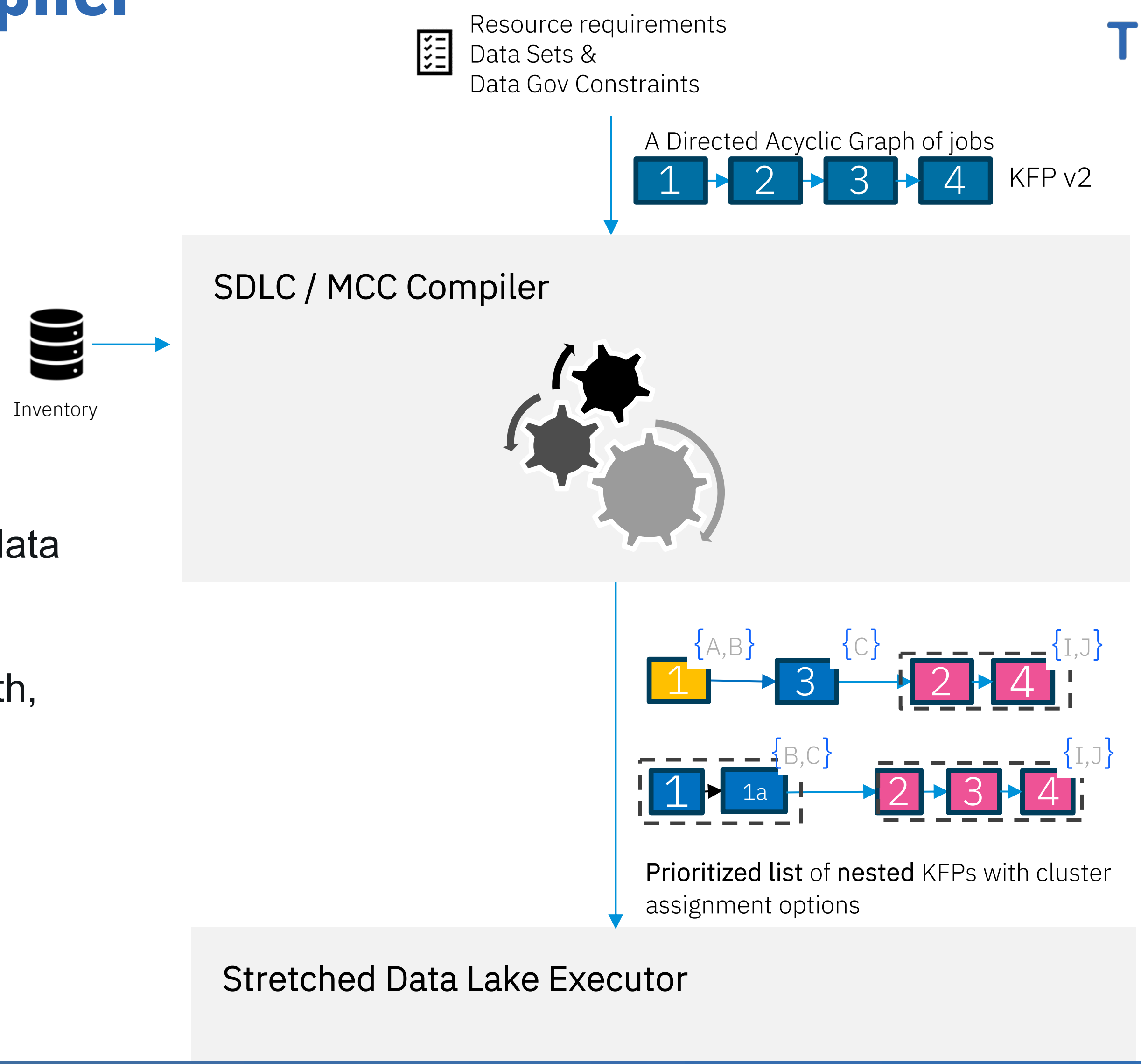


- Kubeflow Pipelines (kfp)
  - Popular open-source framework for orchestration and deployment of data pipelines
  - Declarative and versatile
  - Able to analyze, partition, inject tasks

- Main components are
  - Stretched Data Lake Compiler
  - Stretched Data Lake Executor

Stretched Data Lake Compiler

- Takes as input
  - a Kubeflow Pipeline annotated with associated requirements
  - cluster inventory
  - dataset governance constraints
- Compiles it into an executable multi-cluster/site, multi-cloud deployment plan
- Meets specified requirements (resources, data governance & compliance policies)
- Optimised for NFRs such as cost, bandwidth, time, sustainability



Kubestellar

Traditional Kubernetes distributions (Kubernetes, EKS, AKS, GKS, OpenShift, k8s, etc.) manage workload configurations **inside** a cluster

kube STELLAR extends Kubernetes to manage workload configurations **across** multiple clusters, across multiple clouds, and in edge locations

Distinguishing features of KubeStellar

- **multi-cluster down-syncing** deploy, configure, and collect status **across pre-existing clusters**
- **up-syncing** from remote clusters (return *any* object, not just status)
- **lightweight logical cluster** support (KubeFlex, kcp, kind, etc.)
- **resiliency** to support disconnected operation and intermittent connectivity

Additional features

- non-wrapped / kubernetes-object-native denaturing (enables hierarchy) (no requirement to wrap objects)
- rule-based customization (grouping) - automate the customization of your deployments
- status summarization - summarize the status returned from all your deployments
- scalability - scale to a large number of objects, overcoming default Kubernetes limitations

source: kubestellar.io

Kubestellar in TEADAL

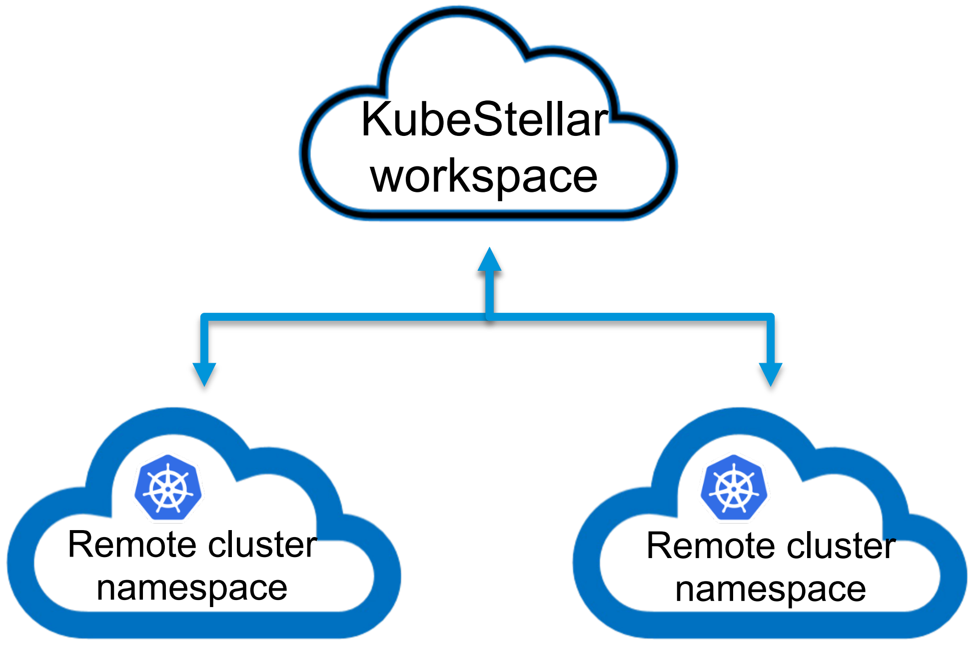
Testbed components run natively in k8s

- i.e., data stores, Airflow, etc..

Pipelines, flows, access GW, etc, will run “in” kubestellar workspace

- Kubestellar will deploy k8s objects in the relevant cluster
- Kubestellar collects k8s statuses and make them available in the “workspace” providing update to the workload

Offloading the task of deciding where to run each (pipeline) processing step to Kubestellar



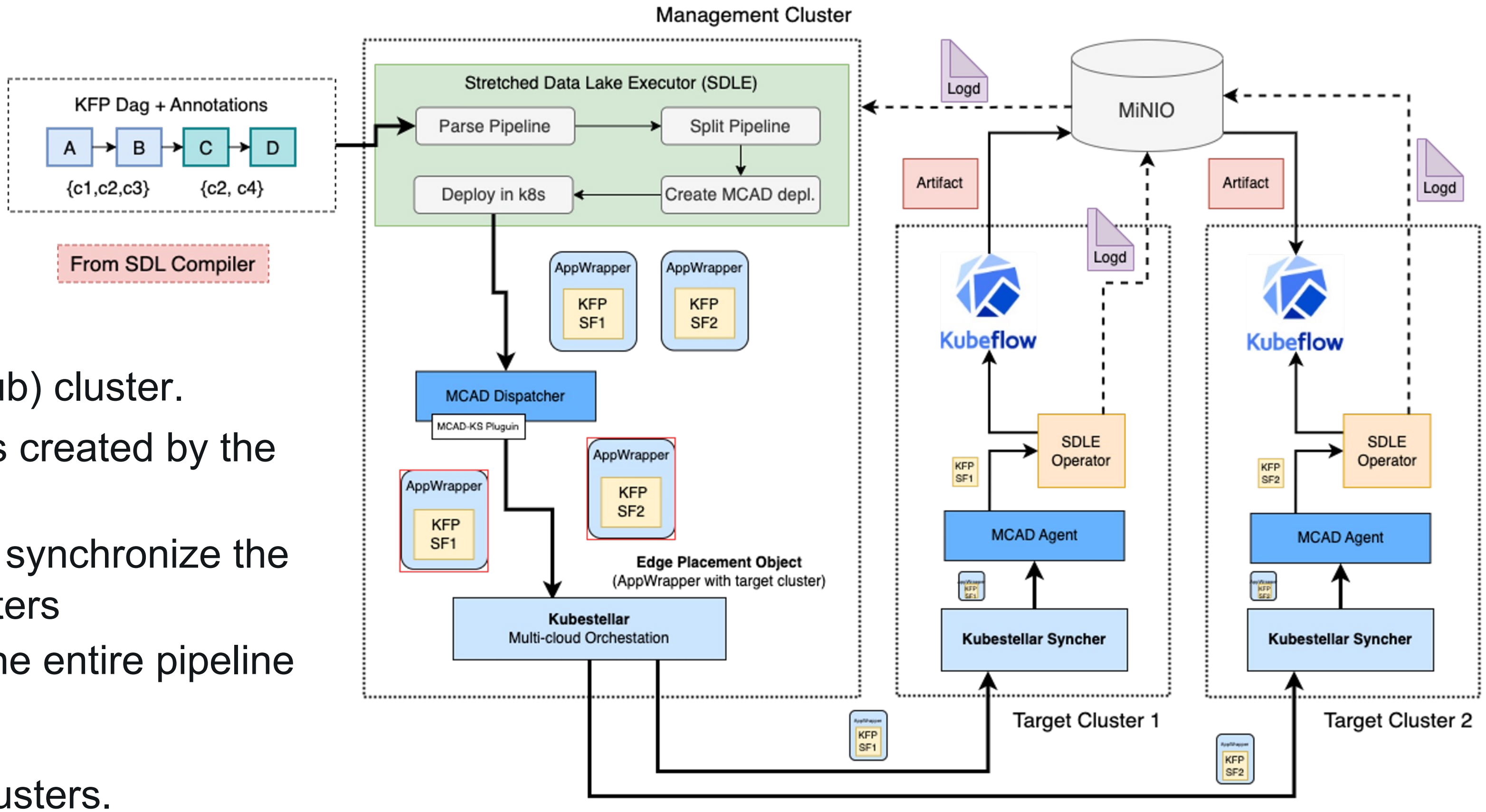
Stretched data lake

Stretched Data Lake Executor

- Takes as input
  - A 2-level kubeflow pipeline
  - Optimized by the compiler

Composed by 2 main components:

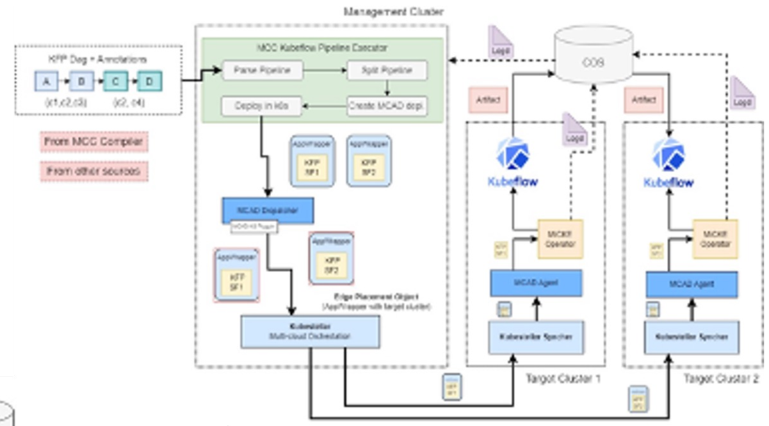
- SDLE Controller:
  - Located in the management (hub) cluster.
  - Splits the pipeline by the groups created by the SDLC.
  - Uses MCAD and Kubestellar to synchronize the deployment with the target clusters
  - Orchestrates the execution of the entire pipeline
- SDLE Operator:
  - Located in the target (spoke) clusters.
  - Monitors the *KubeflowPipeline* k8s objects and submits it to kfp once it receives it into the cluster.



Stretched Data Lake Executor - Internals

Under the hood, Kubestellar synchronizes the MCAD object with the target cluster specified in the MCAD description.

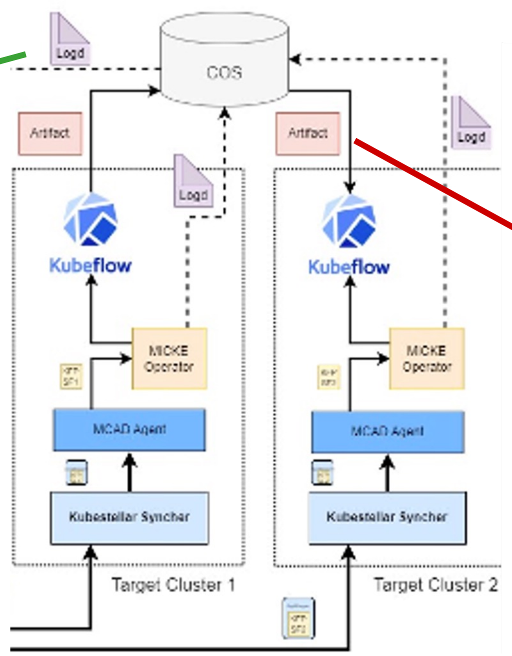
In the target cluster (Spoke), the MiCKE Operator is responsible for reading the object received by KS+MCAD, and it submits and executes the *grouped-pipeline* contained within it in the local Kubeflow Pipelines (KFP) deployment.



The MiCKE Operator is monitoring the pods being created by KFP. With this information it stores **2 different objects** in the object store for each task in the *grouped-pipeline*:

-The first is a 0 bytes objects created when the task starts.

-The seconds is a file containing the logs, stored when the task finishes.



The **Artifacts** (data) created by one *grouped-pipeline*, that have to be read by another *grouped-pipeline*, are stored and thus transferred using an external object store.

Meet the TEADAL Consortium

THANKS

TEADAL.EU

@TEADAL_eu

@TEADAL

Funded by
the European Union

TEADAL project is funded by the EU's Horizon Europe programme
under Grant Agreement number 101070186