Aalto University
School of Science

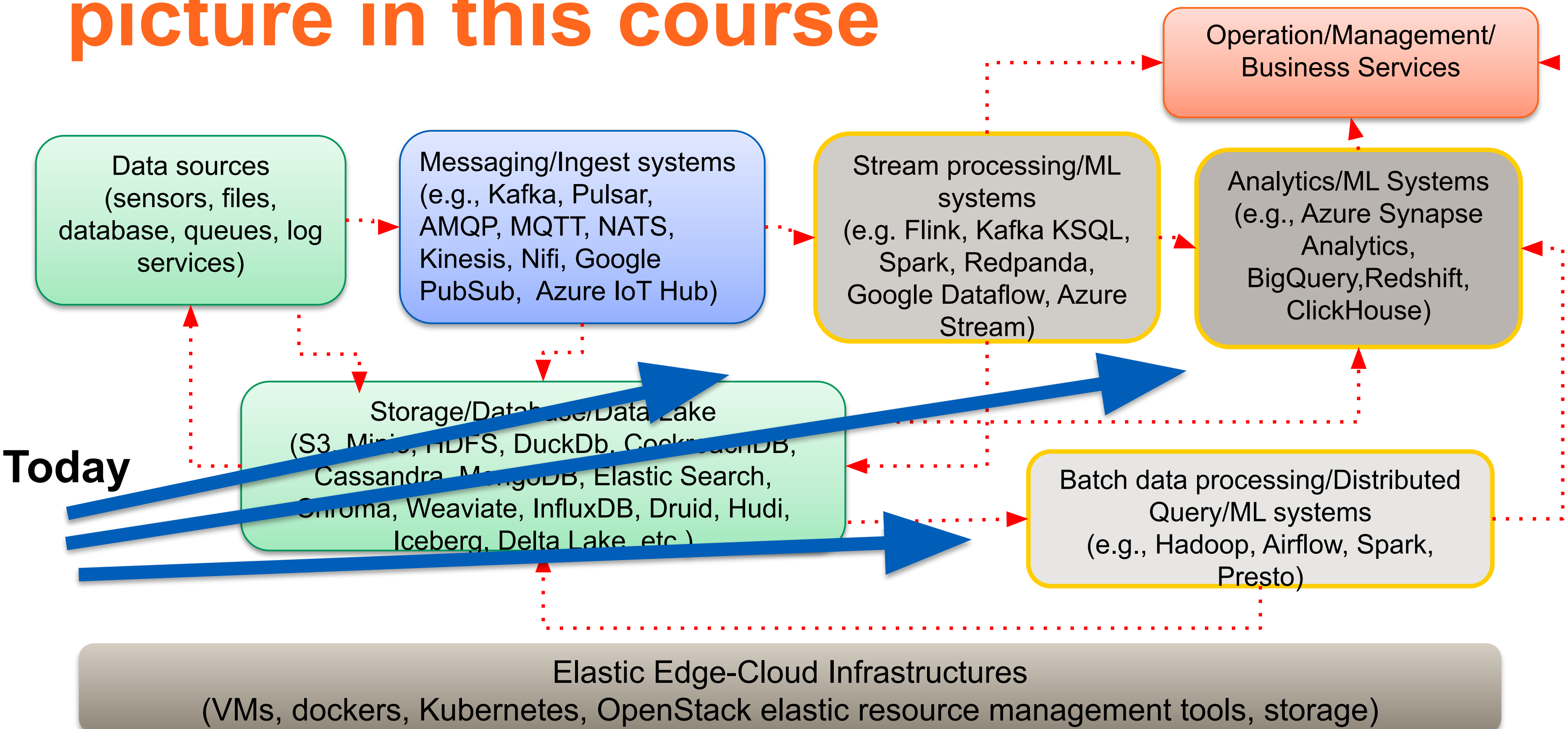
Workflows for Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

Learning objectives

- **Understand the role and use cases of workflows in big data platforms**
- **Understand key concepts and techniques in workflows and able to design workflows**
- **Able to apply common workflow technologies for practical work**

Big data at large-scale: the big picture in this course



Tasks in big data platforms

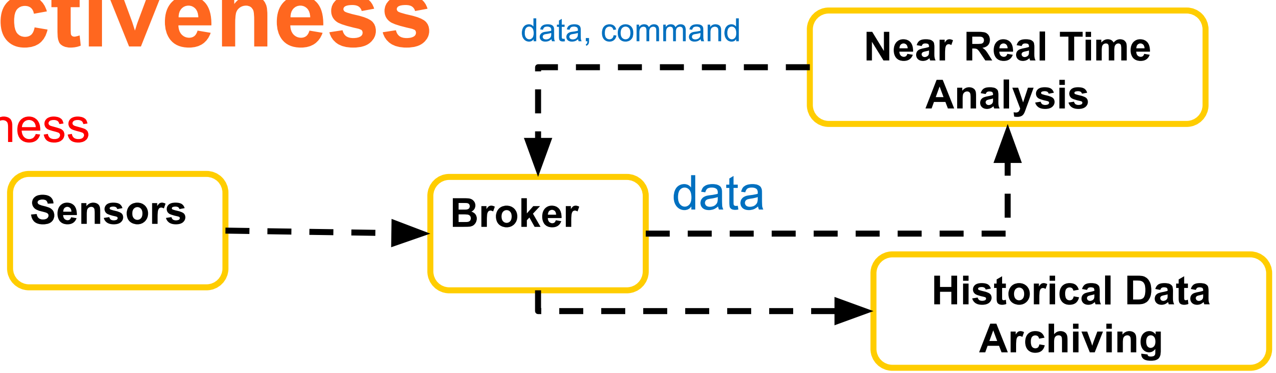
- **Loosely coupled data processing related tasks**
  - data collection and transformation
    - *data transfers, extraction, transformation,*
  - data processing, including machine learning
    - *data analytics, training, serving machine learning algorithms*
- **Loosely coupled platform automation tasks**
  - service deployment, resource elasticity, and backup/recovery
- **Business tasks integration with big data analytics**
  - integration with customer services, bringing insights from data analytics to business decision making

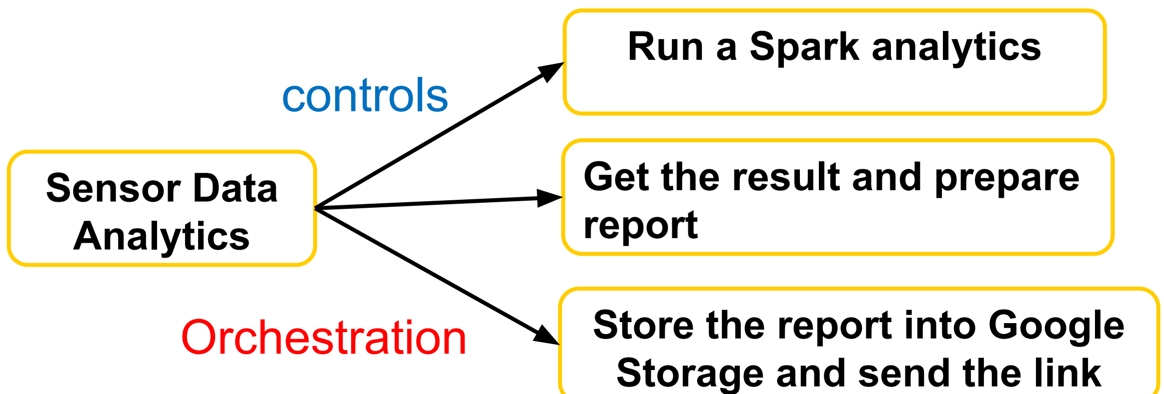
Complex and diverse use cases

- **Deployment and configuration**
  - preparing execution environments and platform components
- **ETL, data cleansing and backup**
  - access and coordinate many different compute services, data sources, ingestion and extraction applications
- **Predictive maintenance**
  - coordination of machine learning pipelines and communication with humans/optimization services
- **Analytics-as a service**
  - metrics understanding, user activities analytics, and customer understanding

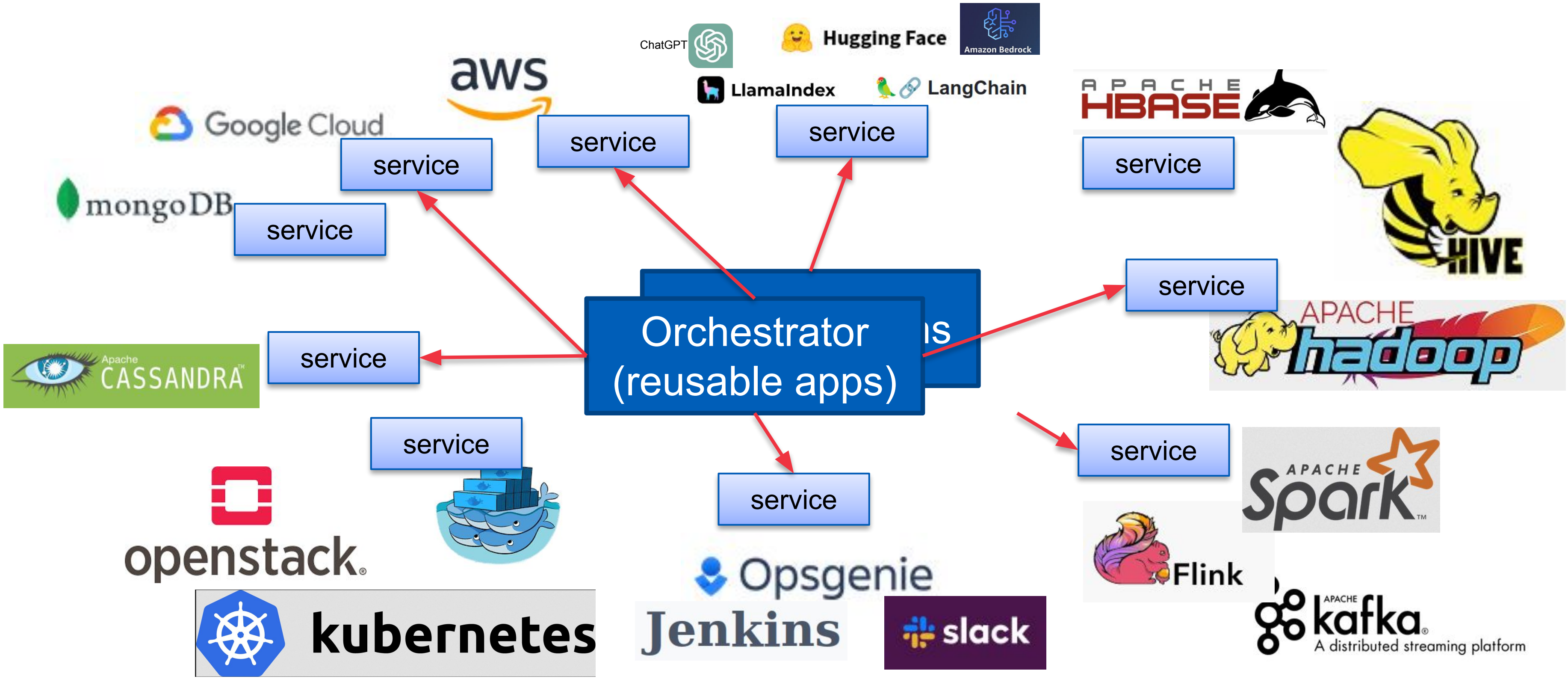
Recall: Orchestration and Reactiveness

Reactiveness





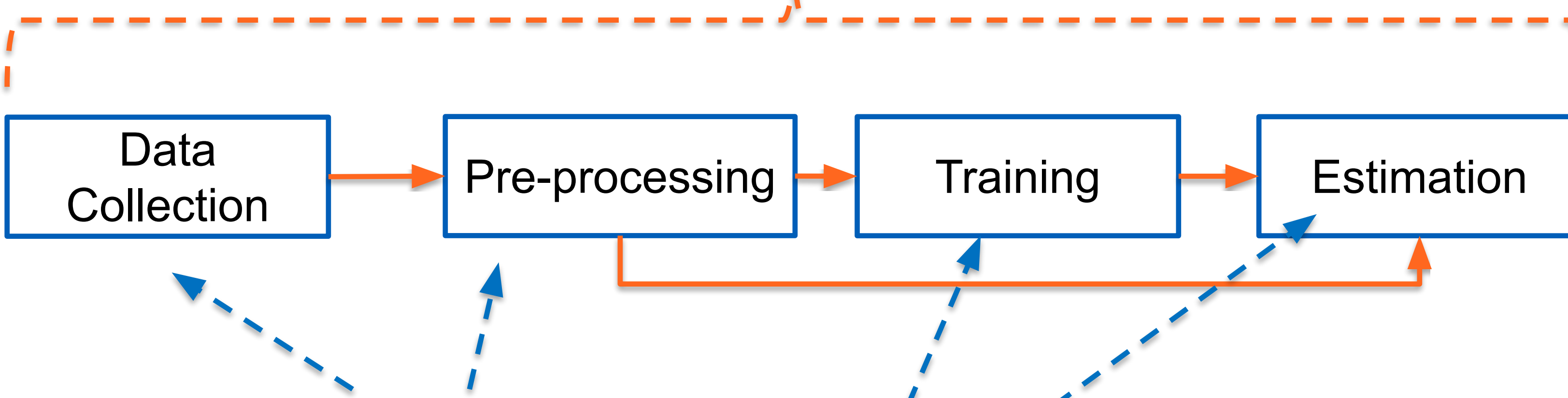
Service orchestration in data platforms: more than just with “data processing”



(Meta) coordination

- **coordination within and across each phase: pipeline/workflow or other types of programs**

(Meta) coordination across sub phases/systems



Coordination within: tasks of workflows, function-a-as-service, Apache Spark,….

Workflows

A workflow specifies a process

- consists of a set of connected tasks/activities
- has tasks/activities carried out by diverse types of software services or humans, each performs a function
- can be automated with/without human intervention
- has data/control task dependencies
- can be reusable (tasks, part of the workflow, and the whole workflow)

Workflow technologies

- **Given many services offering different capabilities, we can combine them for different cases**
  - orchestration of capabilities from different services as the key!
  - reuse/customization of capabilities with a given set of services

- **Workflows are flexibly defined and changed**
  - individual services cannot be changed easily
  - but there are many ways to combine such services!
  - the integration is loosely coupled

We have many workflows that are built in a flexible way for different goals

How to build the workflows and orchestrate tasks in these workflows?

Tasks and workflows

- **Diverse types of tasks**
  - task can be simple or complex (e.g., a task running an AI inference)
  - tasks are performed by software and humans
    - including IoT devices, robots, LLMs and AI Agents
- **Workflow**
  - coordinate/orchestrate many tasks:
    - *the function of tasks* ***is not*** *really "carried out/executed" by workflows* ⇒ *orchestration/coordination*
  - workflow can be simple, like a pipeline of a sequence of tasks or complex with many forks/loops of tasks

Workflow and pipeline/data workflow

- **Data workflow ⇒ data pipeline**

  ”a pipeline is a set of data processing elements connected in series, where the output of one element is the input of the next one”

  Source: https://en.wikipedia.org/wiki/Pipeline_%28computing%29

- **Two interpretations in practice:**
  - a pipeline is a workflow with a simple structure
  - a pipeline coordinates different (sub)workflows
- **Note: sometimes the “data pipeline” here is just an abstract design**

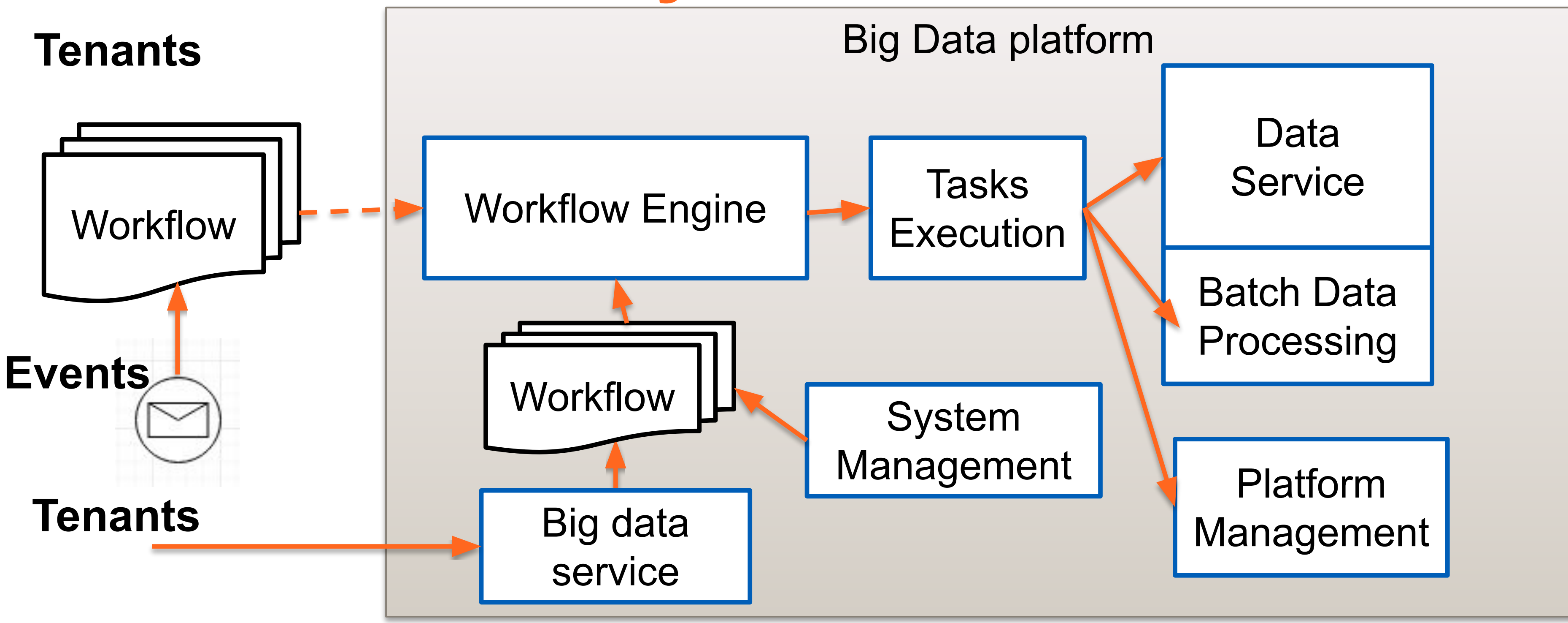
Types of workflows

- **Business workflows/processes**
  - business processes in enterprise computing (e.g., BI, ERP, and e-commerce)
- **Scientific workflows**
  - in scientific computing and high performance computing (e.g., bioinformatics, astrophysics, material science simulations)
- **DevOps workflows**
  - at system level for automating infrastructure provisioning, system recovery, software testing, etc.

Key components

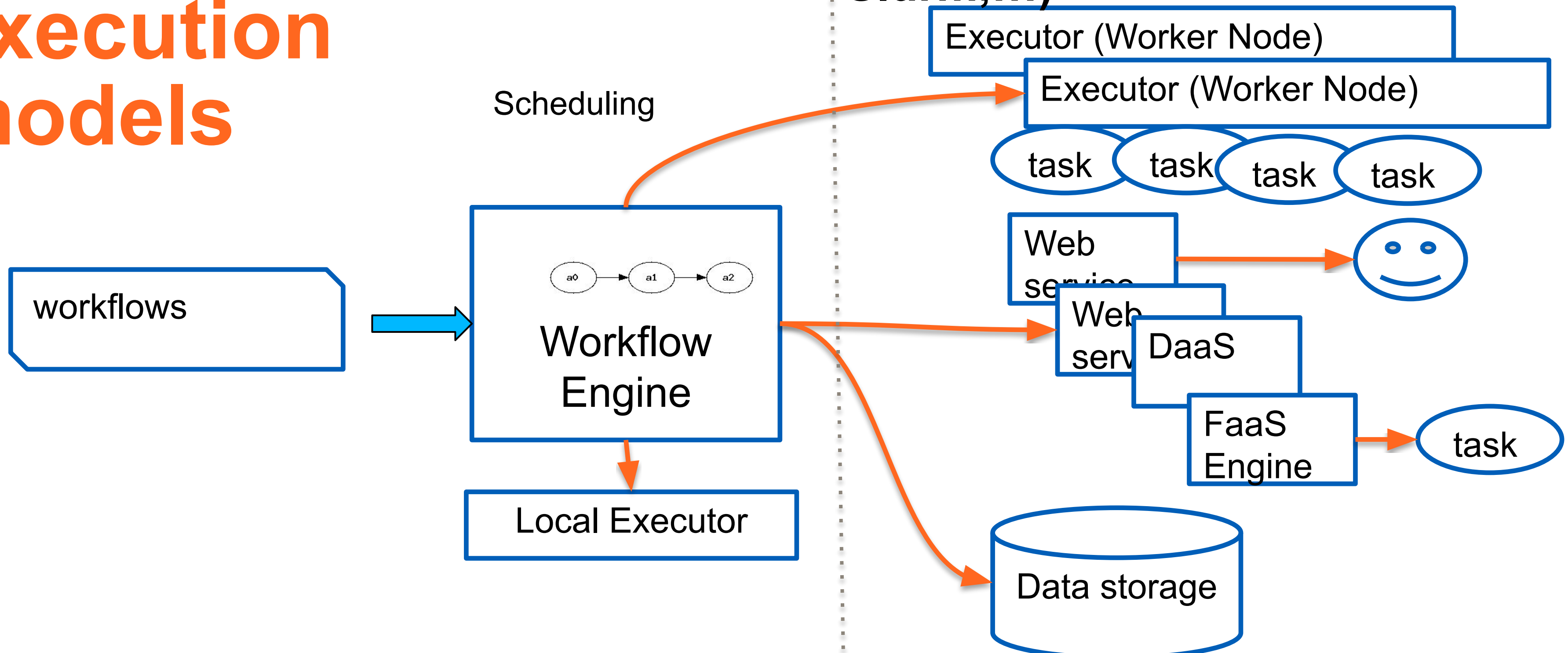
- **Tasks/activities**
  - describe a single work (it does not mean small)
  - tasks can be carried out by humans, executables, scripts, batch applications, stream applications and other types of services.
- **Workflow languages**
  - structure/describe tasks, dataflows, and control flows
- **Workflow engines**
  - execute the workflow by orchestrating tasks
  - usually *call remote services* to run tasks

Workflows in big data platforms: more than analytics

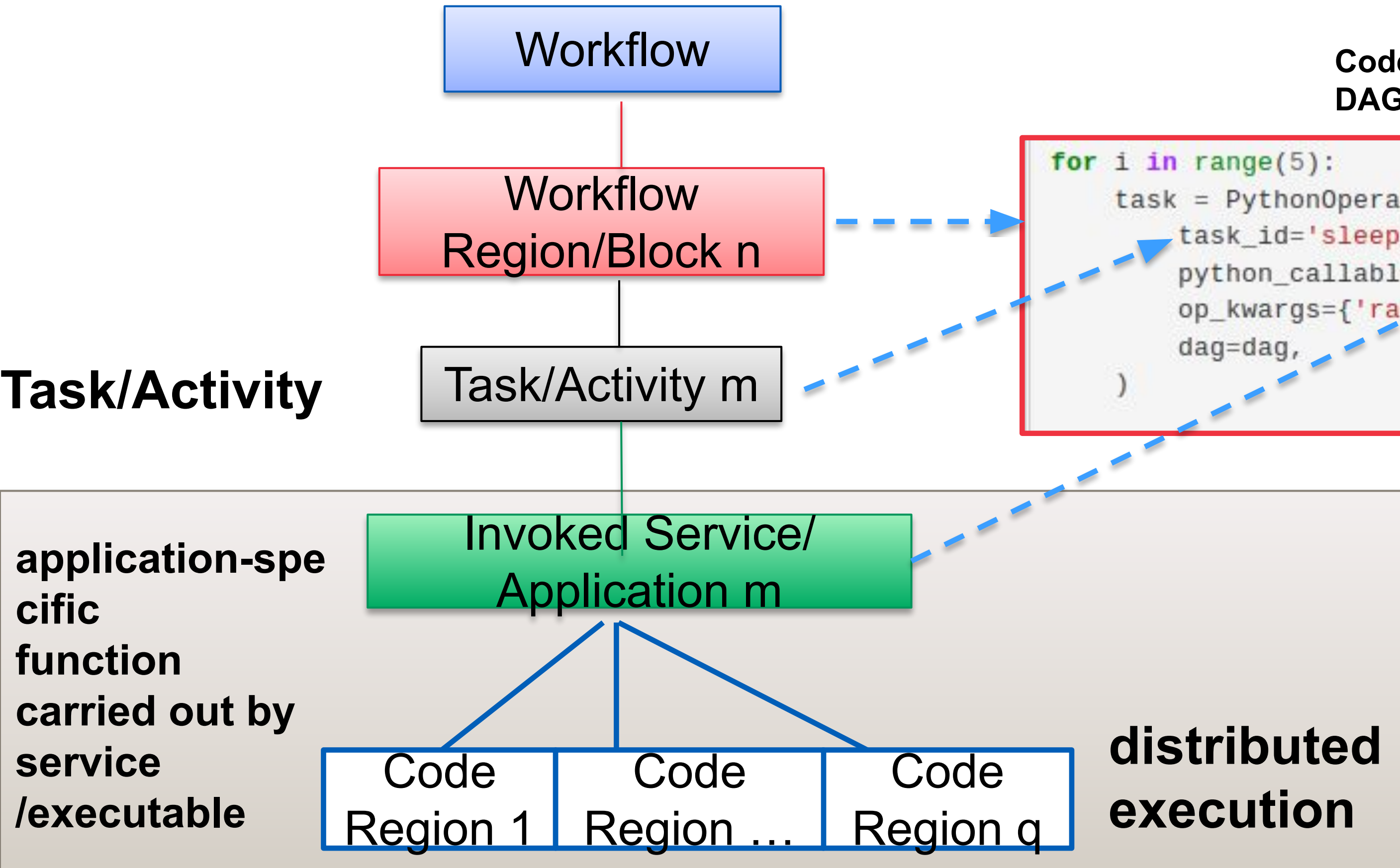


Common workflow execution models

Major works are carried out in distributed nodes (manage by Kubernetes, Celery, Dask, Ray, Slurm,…)



Structured view of workflows



Code captured from Airflow:
DAG: example_python_operator

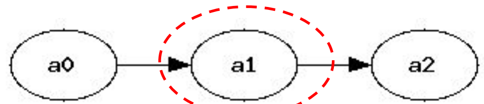
```
for i in range(5):
    task = PythonOperator(
        task_id='sleep_for_' + str(i),
        python_callable=my_sleeping_function,
        op_kwargs={'random_base': float(i) / 10},
        dag=dag,
    )
```

Note: the invoked service/application can be complex (e.g., another workflow)

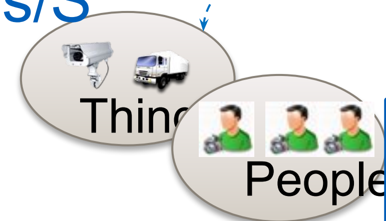
Describing workflows

- **Programming languages with procedural code**
  - general- and specific-purpose programming languages, such as Java, Python, Swift
  - common ways in big data platforms for data analytics and system automation
  - data processing workflows: common in data science and data analysis
- **Descriptive languages with declarative schemas**
  - BPEL, YAML, JSON and several languages designed for specific workflow engines
  - common in business and scientific workflows

Task dependencies & orchestration

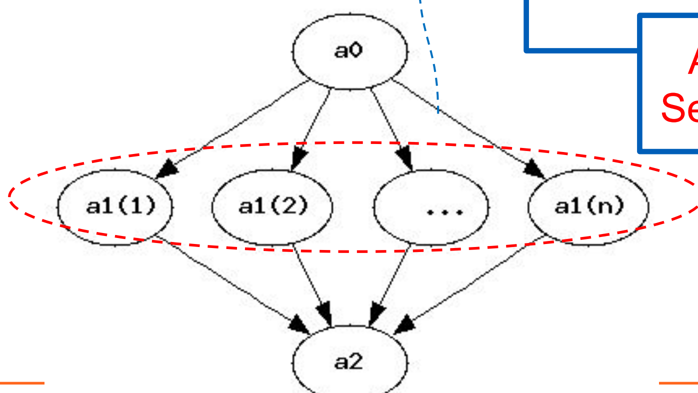


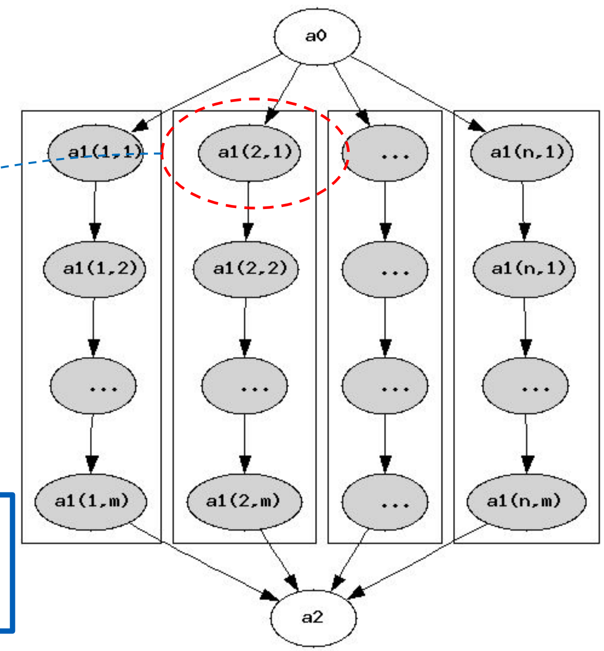
Executors/Services



Data Analytics

AI/ML/LLM Service/Agent



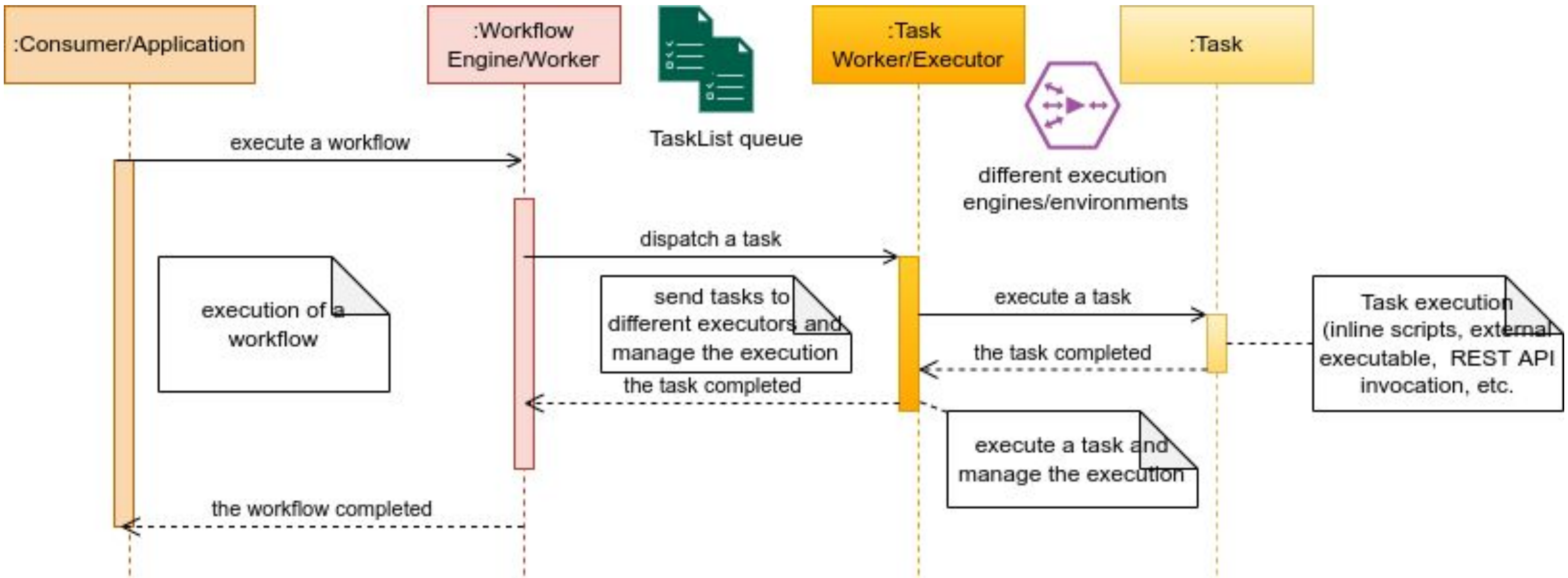


Structures of the workflow: DAG (Directed Acyclic Graph) is a common model

Modern data workflows in clouds

- **Invoked services/applications**
  - cloud services: with diverse types of APIs and protocols
  - serverless functions
  - AI/ML/LLM agents/services
- **Workers/Executors**
  - distributed workers in distributed compute nodes of data centers
    - *interfaces via APIs and messaging systems*
  - containers and distributed task executors
  - cloud orchestration
- **Python-based for data analytics and ML workflows**
  - mainly for data transformation and analysis

Complex execution model across distributed machines



Challenges in managing dependencies, failures and performance

Runtime aspects

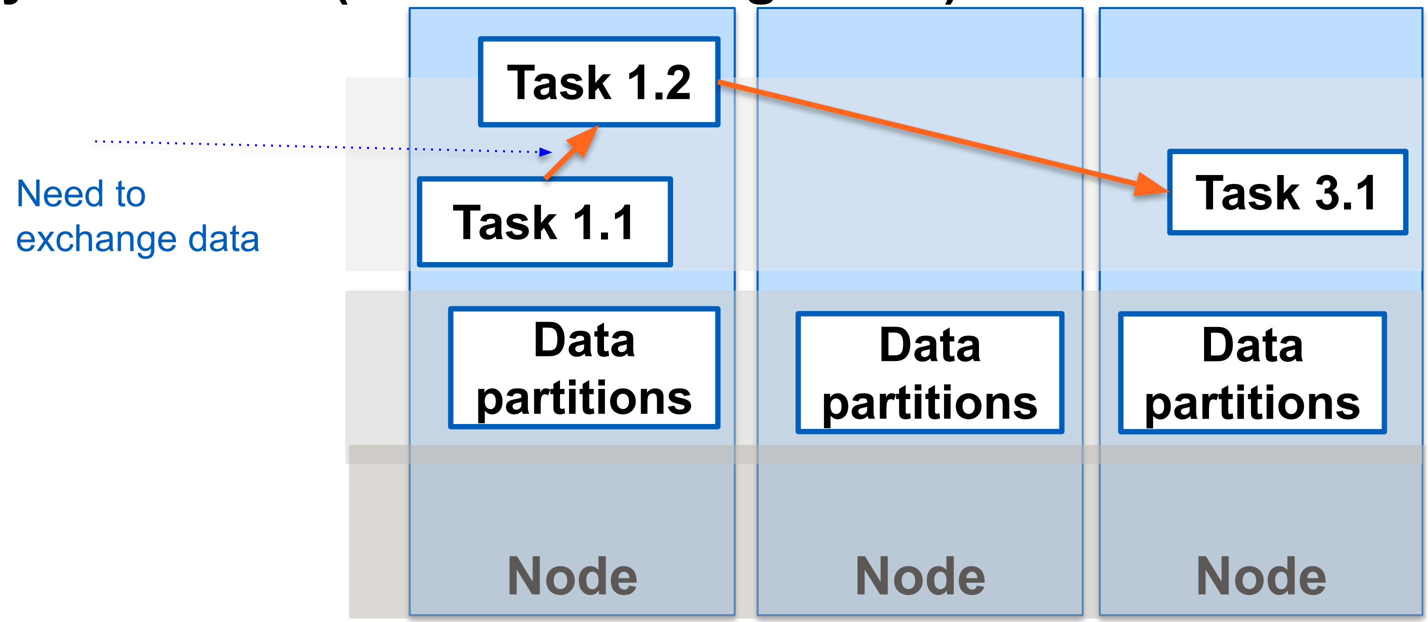
- **Parallel and distributed execution**
  - tasks are executed in different machines (by external invoked applications/services) ⇒ multiple running workflows in the same system
- **Long and/or periodic running**
  - can be hours or weeks! ⇒ pausing and resuming workflows are normal
- **Checkpoint and recovery**
  - dealing with failures at different levels: workflows and tasks retry/recovery
- **Monitoring and tracking**
  - States and performance metrics: queuing, running, idle, suspended, failed
- **Stateful management**
  - dependencies among tasks w.r.t control and data, stateful tasks ⇒ global services for managing states and data among tasks

Select/build workflows in your platforms

- **Rich data services**
  - for data storing/retrieving tasks
- **Big data computation engines**
  - for data processing tasks with different workloads: ML and (batch/stream) big data processing
- **Different underlying cloud/distributed computing infrastructures**
  - for resource management tasks and workflow infrastructures
- **REST APIs and messaging systems integration**
  - for widely integration with other services (e.g., business services)

Recall: exchange data possibilities

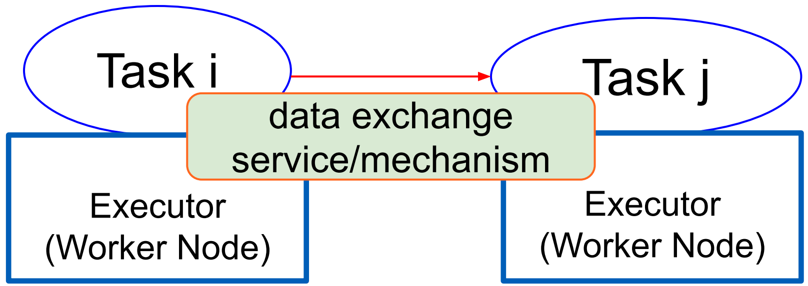
Try to analyze and identify this problem with a framework of your choice (and fault management)



State and exchange data

- **How do tasks exchange data?**

By **value or by reference** exchange?



Possibilities:

- shared file systems
- global/common storage systems
- key/value database
- external processes for pulling data

Some important aspects:

- Asynchronous task execution in which the task results must be retrieved with a different method
- Idempotency (retries, fault-tolerance) and caching
- SerDe (serialization and deserialization)
- Performance of shared systems/services for data exchange
- Deployment consideration

Select/build workflows in your platforms

- **Scheduling**
  - scheduling in a large resource pool (e.g., using clusters)
- **Elasticity**
  - elasticity controls of virtualized resources (workers/executors, e.g. VMs/containers/Kubernetes) for executing tasks
- **Multiple levels of concurrency/parallelism**
  - cluster level vs node level

- **Examples**
  - periodic cron schedules, backfill, opportunistic schedules
  - increase number of distributed workers/cluster sizes
  - heterogeneous resources for tasks: lightweight compute nodes & high-end nodes

Wu, F., Wu, Q. & Tan, Y. Workflow scheduling in cloud: a survey. J Supercomput 71, 3373–3418 (2015). https://doi.org/10.1007/s11227-015-1438-4

Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama. 2019. A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends. ACM Comput. Surv. 52, 4, Article 68 (August 2019), 36 pages. https://doi.org/10.1145/3325097

Monitoring

- **Understand the states of tasks and the whole workflow**
  - workflow level vs task/activity level vs invoked service/application level
- **Multiple levels of instrumentation and monitoring**
  - Workflow Engine/Worker
  - Scheduler
  - Task worker/executor

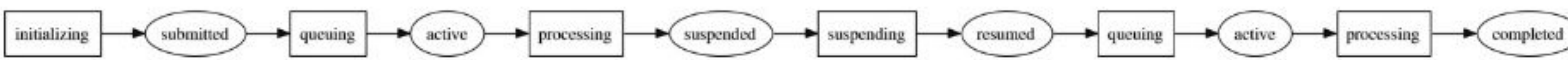
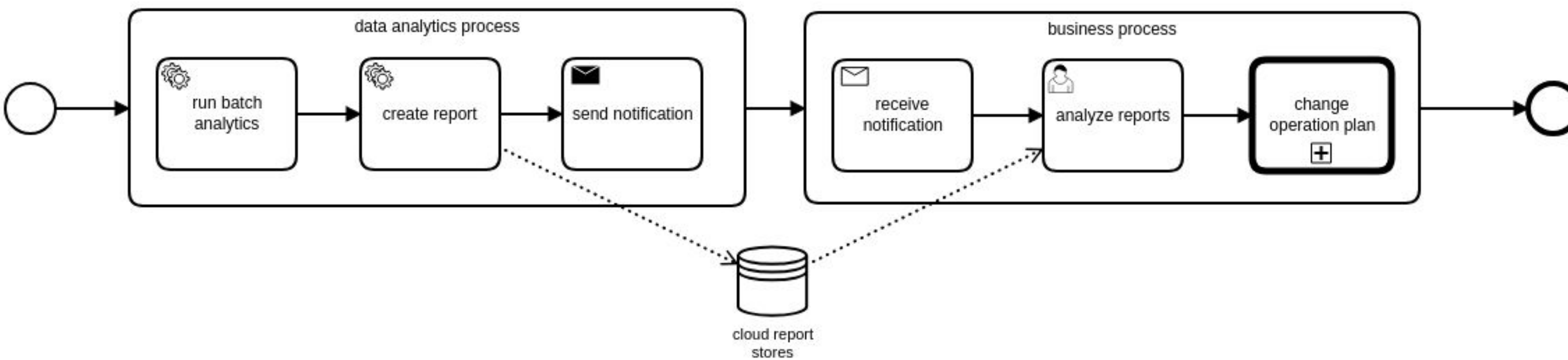


Fig. 3. Discrete process model of the tracing execution of an activity. □ represents an execution phase, ○ represents an event.

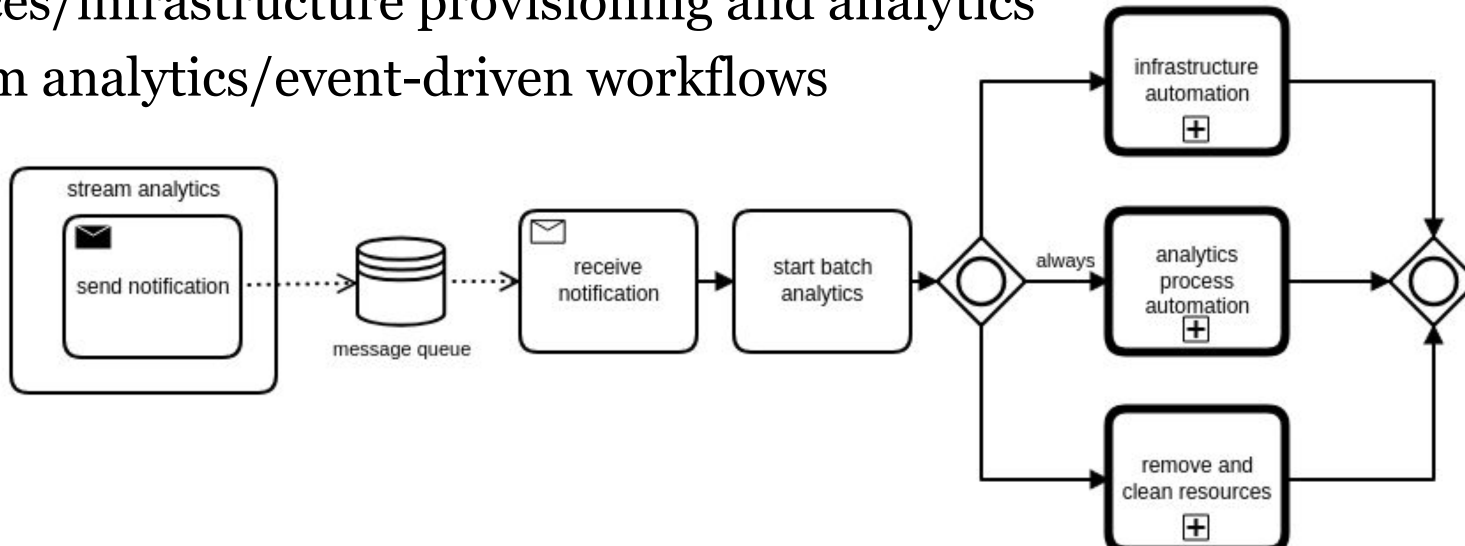
Select/build workflows in your platforms

- **Integration**
  - data analytics processes and business processes
  - include human-in-the-loop



Select/build workflows in your platforms

- **Integration**
  - multiple types of workflows for services/infrastructure provisioning and analytics
  - stream analytics/event-driven workflows



Existing frameworks for your study

- **Apache Oozie**
  - designed to work with Hadoop: orchestrating Hadoop jobs
- **Serverless-based: Function-as-a-Service**
  - e.g., Microsoft, Google, AWS serverless/function-as-a-service
- **Apache Airflow: a generic workflow framework (hands-on)**
- **Argo Workflows**
  - Container-native workflow engine
- **Uber Cadence (https://cadenceworkflow.io) & Camunda (https://camunda.com/)**
  - Connecting to business activities + human in the loop
- **Serverless Workflow: domain-specific language FaaS centric**
  - https://serverlessworkflow.io/

Example with Apache Airflow

https://airflow.apache.org

Airflow overview

- **Originally from Airbnb**
- **Features**
  - dynamic, extensible, scalable workflows, programmable language-based workflows
    - *workflows written as procedural code*
- **Good and easy to study to understand concepts of workflows:**
  - suitable for data analytics/ETL, ML engineering and DevOps automation

Many connectors

- Airbyte
- Alibaba
- Amazon
- Apache Beam
- Apache Cassandra
- Apache Drill
- Apache Druid
- Apache HDFS
- Apache Hive
- Apache Kylin
- Apache Livy
- Apache Pig
- Apache Pinot
- Apache Spark
- Apache Sqoop
- Asana
- Celery
- IBM Cloudant
- Kubernetes
- Databricks
- Datadog
- DBT cloud
- Dingding
- Discord
- Docker
- Elasticsearch
- Exasol
- Facebook
- File Transfer Protocol (FTP)
- Github
- Google
- gRPC
- Hashicorp
- Hypertext Transfer Protocol (HTTP)
- Influx DB
- Internet Message Access Protocol (IMAP)
- Java Database Connectivity (JDBC)
- Jenkins
- Jira
- Microsoft Azure
- Microsoft PowerShell Remoting Protocol (PSRP)
- Microsoft SQL Server (MSSQL)
- Windows Remote Management (WinRM)
- MongoDB
- MySQL
- Neo4J
- ODBC
- OpenFaaS
- Opsgenie
- Oracle
- Pagerduty
- Papermill
- Plexus
- PostgreSQL
- Presto
- Qubole
- Redis
- Salesforce
- Samba
- Segment
- Sendgrid
- SFTP
- Singularity
- Slack
- Snowflake
- SQLite
- SSH
- Tableau
- Telegram
- Trino
- Vertica
- Yandex
- Zendesk

From https://airflow.apache.org/docs/

Cloud integration and big data support

- **Several supports with known cloud providers**
  - Microsoft Azure
  - Amazon Web Services
  - Databricks
  - Google Cloud Platform (Google Composer)
- **Big data supports**
  - Hadoop, Hive, Druid, Presto
- **Distributed execution**
  - Celery, Dask, and Kubernetes

Airflow workflow structure

- **Workflow is a DAG (Direct Acyclic Graph)**
  - a workflow consists of a set of activities/tasks represented in a DAG
  - workflow and activities are programed using Python
    - the workflow structures described in code
- **Workflow activities are described by Airflow operator objects**
  - tasks are created when instantiating operator objects

Airflow operators/tasks

- **Tasks are implemented using operators**
- **Rich set of operators**
  - we can program different kinds of tasks and integrate with different systems
- **Different types of operators for workflow activities**
  - BashOperator, PythonOperator, EmailOperator, SimpleHttpOperator, BaseSQLOperator, BaseSensorOperator, DockerOperator, HiveOperator, SparkSubmitOperator,SageMakerTrainingOperator, PrestoToMysqlOperator, SlackAPIPostOperator
- **Remember:**
  - such operators will be executed by corresponding services

Example for uploading state logs

```
t_download_data =  PythonOperator(
    task_id="download_data",
    python_callable=download_data,
    op_kwargs={'source_file':source_file, 'dest_file':temp_dest_file},
    dag=dag,
    )
#the dest file from the download task will be used for anal
t_basic_aggregration = PythonOperator(
    task_id='alarm_analytic',
    python_callable=basic_aggregation,
    op_kwargs={'input_file':temp_dest_file,'report_destinat
    dag=dag,
    )
t_uploadgcs =  LocalFilesystemToGCSOperator(
    task_id="upload_local_file_to_gcs",
    src=report_destination,
    dst=gcs_dest_file,
    bucket=GCS_CONF["bucket"],
    gcp_conn_id=GCS_CONF["gcp_conn_id"],
    dag = dag
    )
```

```
t_insert_data_warehouse = PythonOperator(
    task_id='insert_data_warehouse',
    python_callable=data_to_bigquery,
    op_kwargs={'input_data_src':f'file://{report_destination}',
               'table_id':BIGQUERY_CONF["table_id"],
               'project_id':BIGQUERY_CONF["project_id"],
               'credentials':credentials},
    dag=dag)


t_msnotification = PythonOperator(
    task_id='teams_notification',
    python_callable=post_notification,
    op_kwargs={'gcs_report':gcs_file_url,'teams_webhook':teams_webhook},
    dag=dag)


t_clean_data = PythonOperator(
    task_id='data_cleansing',
    python_callable=clean_data,
    op_kwargs={'dest_files':[temp_dest_file,report_destination]},
    dag=dag)
```

In our GIT course (tutorials)

Example

upstream task

```
"""
the dependencies among tasks

now you have to remember how different tasks exchange data:
- they pass data via files and you use a local file system, but
task A and task B are not executed in the same machine
- they pass data via a global data storage, then some upload/download of data
must be implemented.

thus, you have to see the task implementation in detail. This example, basically,
works only for local or file sharing systems as we implement download, check quality,
clean data, etc. using local file systems.
"""

t_download_data >> t_basic_aggregration >> t_uploadgcs >> t_insert_data_warehouse
t_insert_data_warehouse >> t_msnotification  >> t_clean_data

```

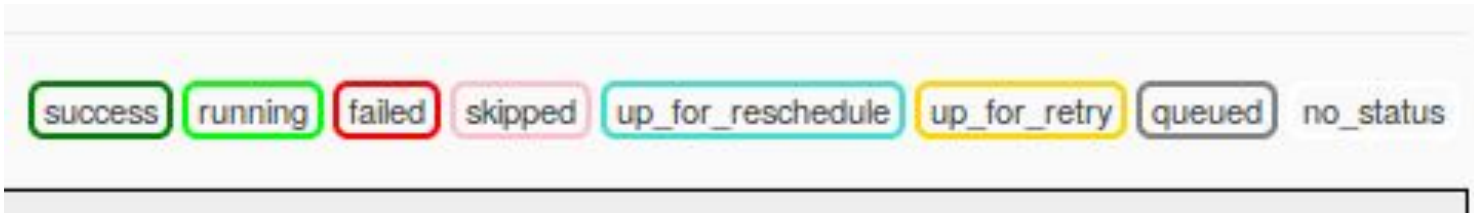
downstream task

Scheduling and execution

- **You can schedule the workflow like a cron job**
  - execute once, every minutes, hours, ...
- **Trigger from external**
  - tasks can be triggered as normal (upstream tasks finishes, dependencies)
  - or specific triggers
- **Very suitable ingestion and batch analytics job managements**
  - the ingestion and analytics are done within tasks
  - schedule based on analytics needs

Task lifecycle

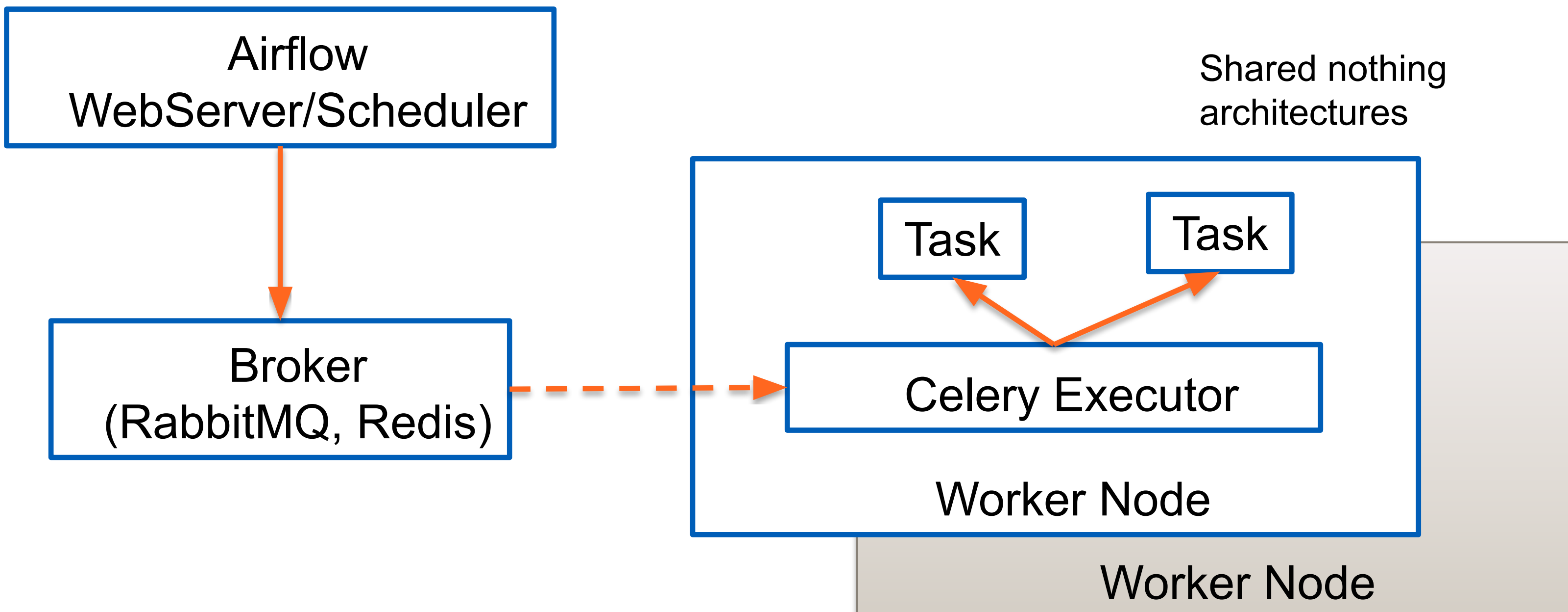
Different states



- **performance metrics can be determined based on states and structures**
- **Interesting in performance analytics?**
  - **Check https://doi.org/10.1016/j.future.2007.01.003**

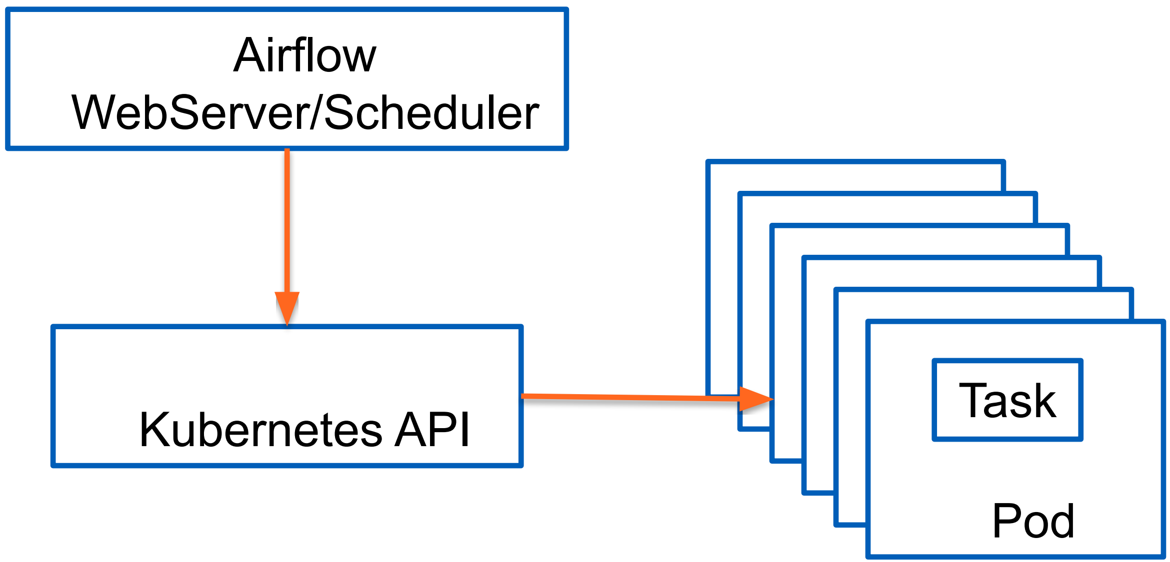
Distributed tasks

You can scale Airflow using workers deployed in different nodes managed by Celery (http://www.celeryproject.org)



Distributed tasks

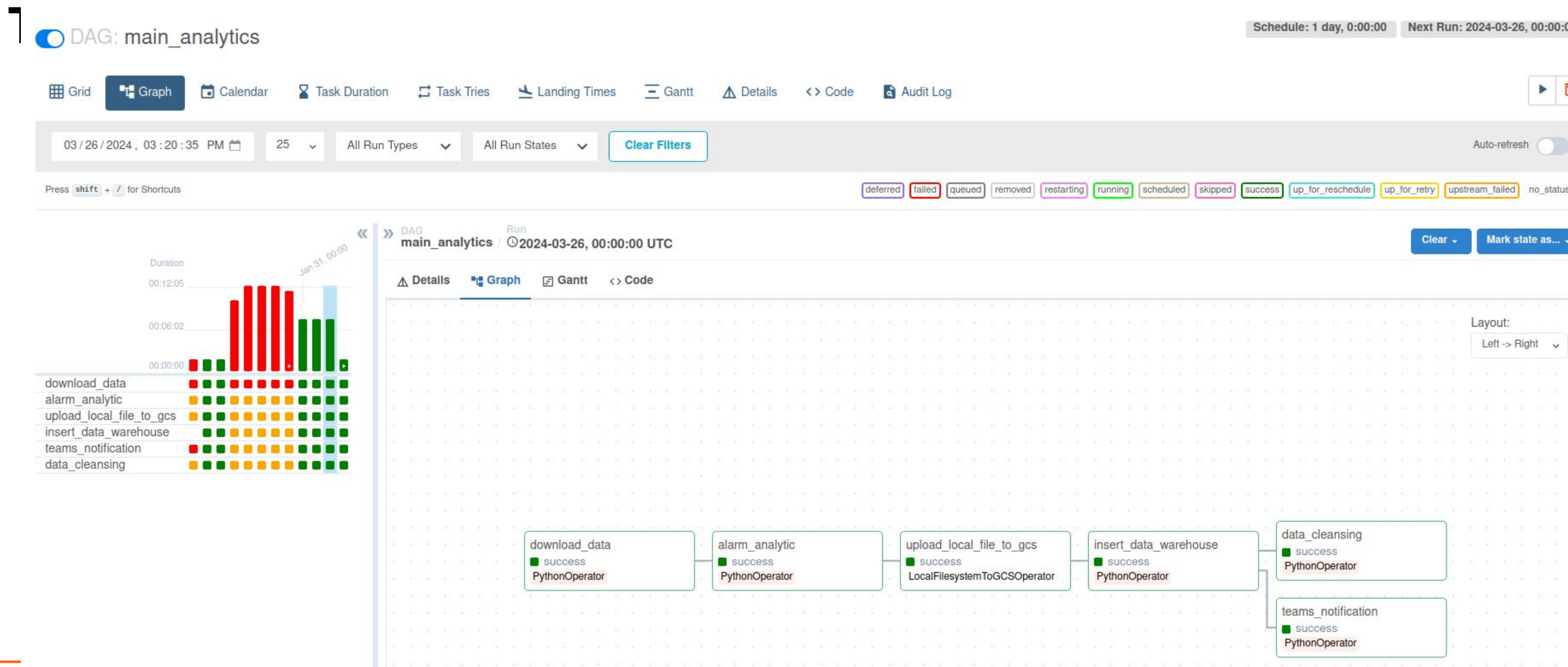
You can scale Airflow to run tasks in Kubernetes



Google Cloud Composer: use Kubernetes

Monitoring UI

download data from various services (via pre-defined API/endpoints) ⇒ analyze data ⇒ store results to Google storage and BigQuery ⇒ send notifications about the result to



Summary

- **Focus:**
  - practical programming with:
    - *Apache Airflow: for data analytics and platform management*
    - *Serverless workflows using function-as-a-service: e.g., AWS Steps*
    - *Kubeflow: for machine learning with big data platforms (if you like ML)*
- **Action:**
  - hands-on and work on concrete examples
    - *try to see if you can implement previous use cases/scenarios in your work with workflows*
  - offering workflows as a service in your platform!
    - *suggest to do some hands-on by configuring and deploying Airflow*

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io