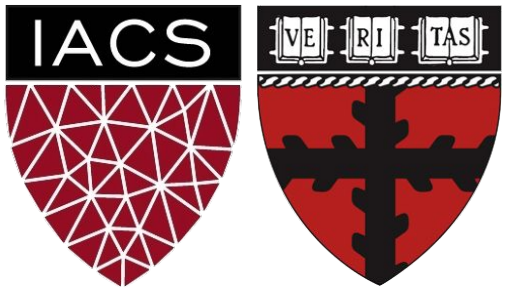
Lecture 2: Data - Dask, Cloud Storage

Advanced Practical Data Science, MLOps

AC215

Pavlos Protopapas
Institute for Applied Computational Science, Harvard



Outline

1. Communication
2. Motivation
3. What are Data Pipelines
4. Dask
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. Task Scheduling
8. Cloud Storage

Outline

1. **Communication**
2. Motivation
3. What are Data Pipelines
4. Dask
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. Task Scheduling
8. Cloud Storage

Communication

- Make sure you have filled out the Survey by Thu 09/09 2PM
- Form project groups and fill this sheet
- Finalize project proposals/approvals

Outline

1. Communication
2. **Motivation**
3. What are Data Pipelines
4. Dask
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. Task Scheduling
8. Cloud Storage

Motivation

The 3 components for better Deep Learning


More Data


Better/Faster Models


Faster Hardware

Motivation

The 3 components for better Deep Learning

More Data

- Storage
- Processing
- Input to Training

Better/Faster Models

- SOTA Models
- Transfer Learning
- Distillation
- Compression

Faster Hardware

- Scaling data processing
- GPU, TPU
- Multi GPU Server Training

Motivation: Data Size

Dataset type	Size range	Fits in RAM?	Fits on local disk?
Small dataset	Less than 2-4 GB	Yes	Yes
Medium dataset	Less than 2 TB	No	Yes
Large dataset	Greater than 2 TB	No	No

Adapted from Data Science with Dask

Motivation: Data Size

Challenges:

- Medium datasets will not all fit in memory (RAM)
- Large datasets will not fit in disk (Hard drive)

Motivation: Data Size

Challenges:

- Medium datasets will not all fit in memory (RAM)
- Large datasets will not fit in disk (Hard drive)

Solution:

- Building data pipelines
  - Read data in batches which can fit in RAM
  - Feed data in batches to GPU
  - Read data from big data store in batches, so not all data need to be present in local hard drive

Motivation: Data Size

Tools:

- Dask
- Google Cloud Storage (Big data store)
- TensorFlow Data
- TensorFlow Records

Motivation: Data Size

Tools:

- **Dask**
- **Google Cloud Storage (Big data store)**
- TensorFlow Data
- TensorFlow Records

} This week

} Next week

Outline

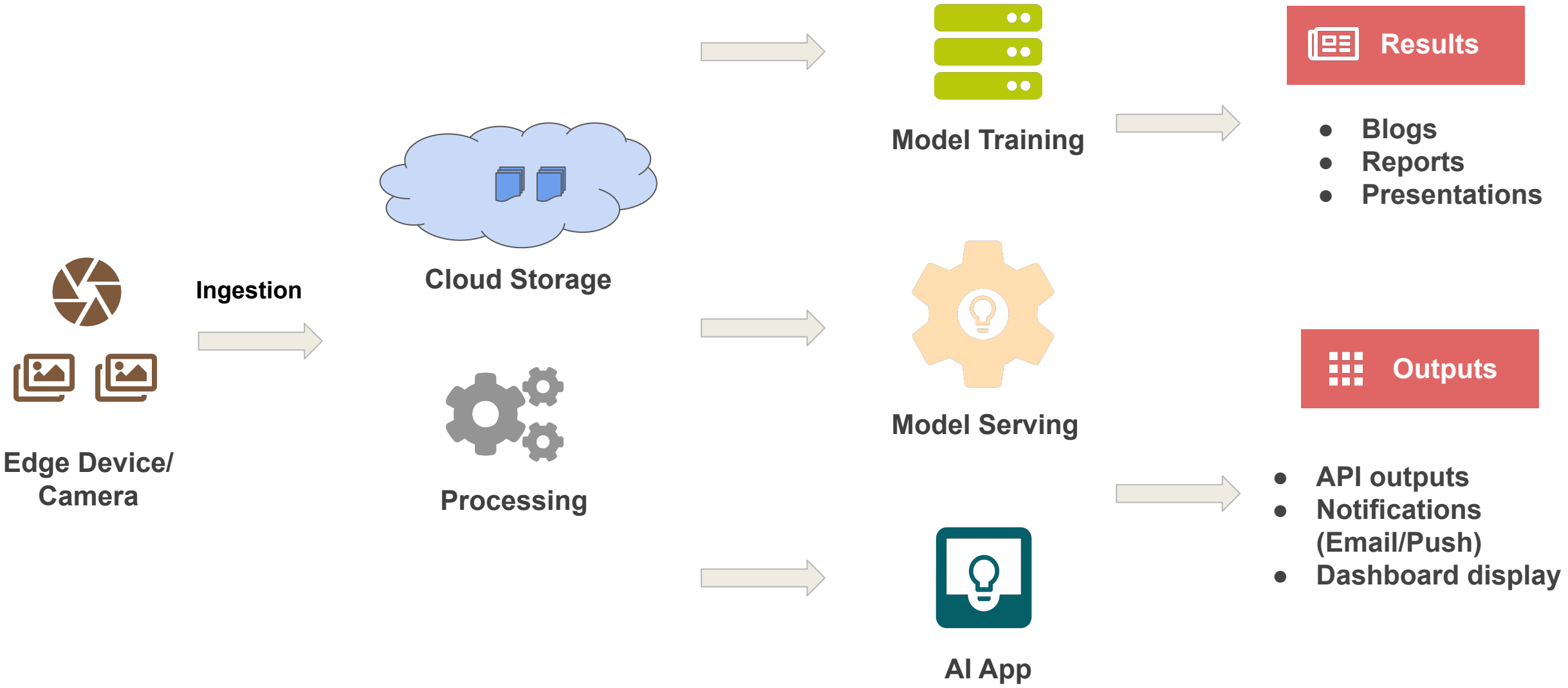
1. Communication
2. Motivation
3. **What are Data Pipelines**
4. Dask
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. Task Scheduling
8. Cloud Storage

What are Data Pipelines

Various data tasks in a Deep Learning project:

- Ingestion
- Preparation
- Pre-processing
- Train, validate, test split
- Pre-process step during model inference

What are Data Pipelines



Outline

1. Communication
2. Motivation
3. What are Data Pipelines
4. **Dask**
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. Task Scheduling
8. Cloud Storage

What is Dask

Dask is a free and open-source library for parallel computing in Python. It allows you to work on arbitrarily large datasets and dramatically increases the speed of your computations.

Pandas

```
# read data using DataFrame API
df = pd.read_csv("path to csv")

print("Shape:", df.shape)
Shape: (100000, 43)

# Display the top rows
df.head()
```

Dask

```
# read data using DataFrame API
df = dd.read_csv("path to csv")

print("Shape:", df.shape)
Shape: (Delayed('int-weyus...'), 43)

# Display the top rows
df.head()
```

Dask's features

What is **unique** about Dask:

- It allows **to work with larger datasets** making it possible to parallelize computation. At the same time, Dask can be used effectively to work with both **medium datasets** on a single machine and **large datasets on a cluster**.

- It **simplifies** the operation and therefore reducing the cost of using more complex **infrastructure**.

Dask

- Dask is fully **implemented in Python** and natively scales NumPy, Pandas, and scikit-learn. It is **therefor easy to learn** for data scientists with a background in the Python (similar syntax) and flexible.

- Dask can be used as a **general framework** for **parallelizing** most Python objects.

- Dask has a very low configuration and maintenance overhead.

Adapted from Data Science with Dask

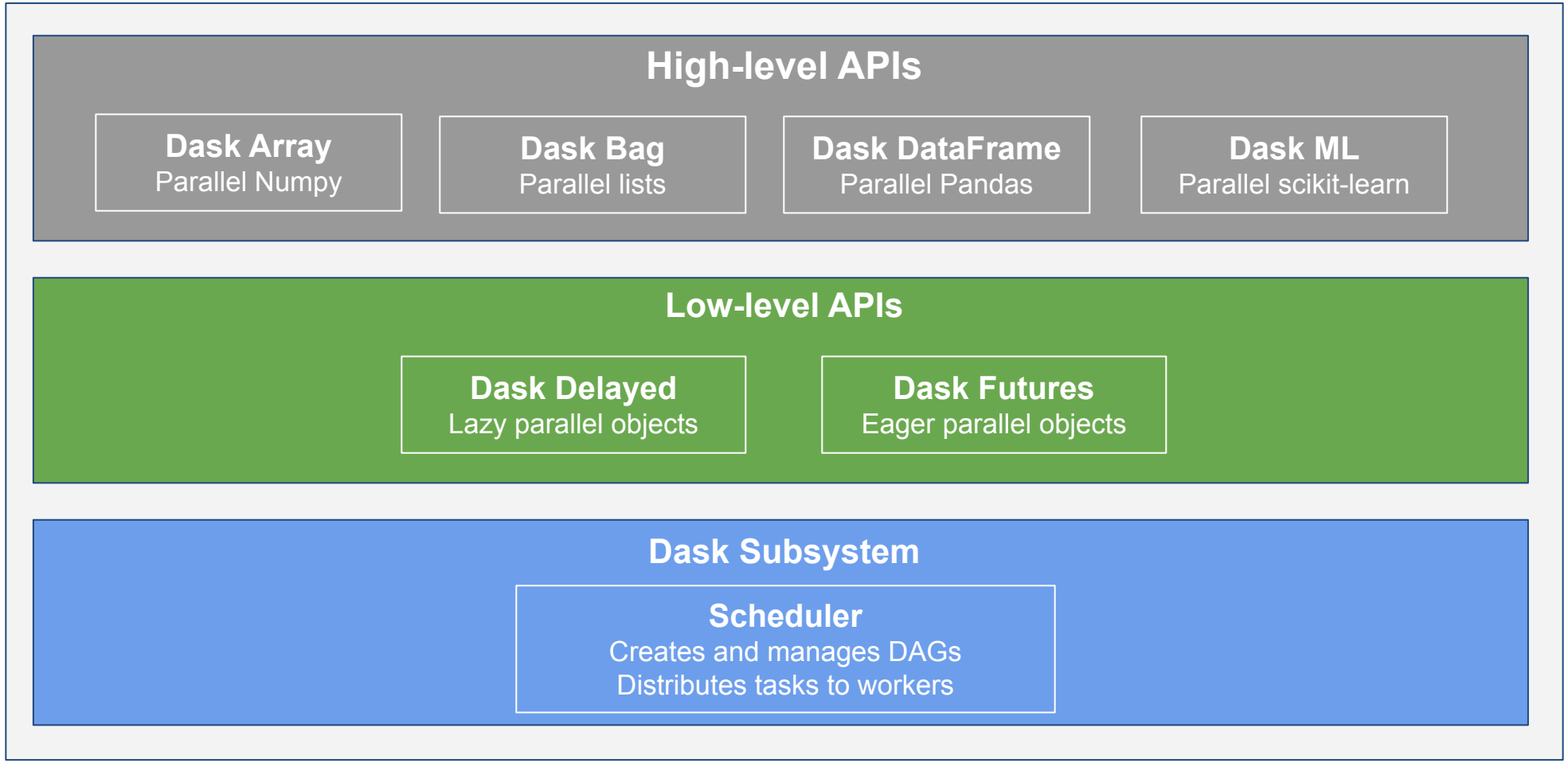
Dask usage

- **Not of great help for small size datasets**: It generates greater overheads. Complex operations can be done without spilling to disk and slowing down process.

- **Very useful for medium size dataset:** it allows to work with medium size in local machine. Difficult to take advantage of parallelism within Pandas (no sharing work between processes on multicore systems).

- **Essential for large datasets**: Pandas, NumPy, and scikit-learn are not suitable at all for datasets of this size, because they were not inherently built to operate on distributed datasets.

Dask Architecture



Adapted from Data Science with Dask

Outline

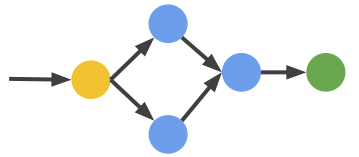
1. Communication
2. Motivation
3. What are Data Pipelines
4. Dask
5. **Directed Acyclic Graph (DAGs)**
6. Computational Resources
7. Task Scheduling
8. Cloud Storage

Directed Acyclic Graph (DAGs)

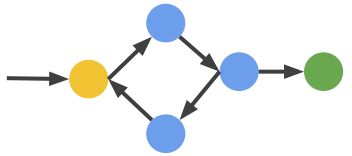
A graph is a representation of a **set of objects that have a relationship** with one another. It is used to representing a wide variety of information.

A graph is consisted by:

- **node**: a function, an object or an action
- **line**: symbolize the relationship among nodes

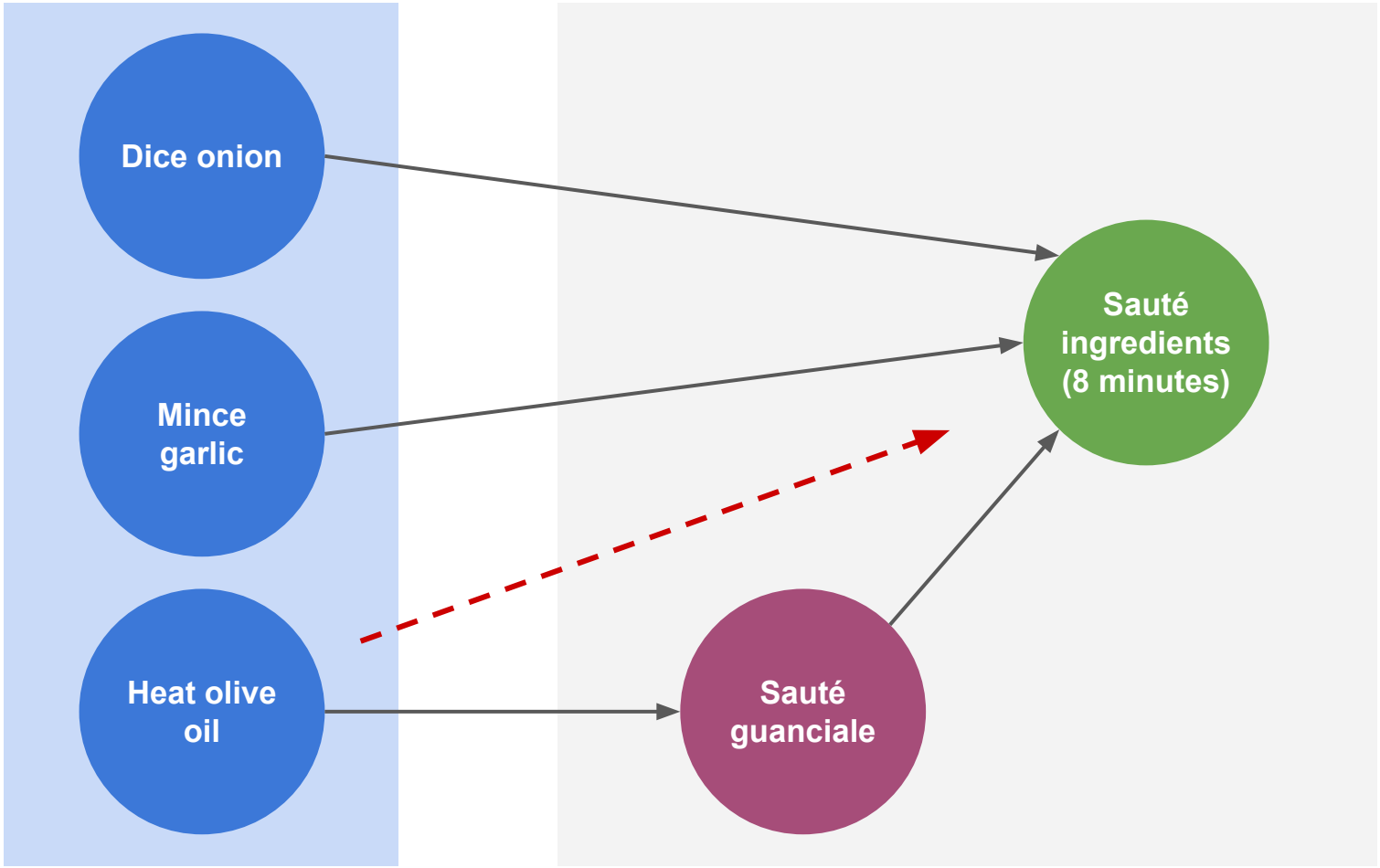


In a directed acyclic graph there **is one logical way to traverse** the graph. No node is visited twice.



In a *cyclic graph*: exist a feedback loop that allow to revisit and repeat the actions within the same node.

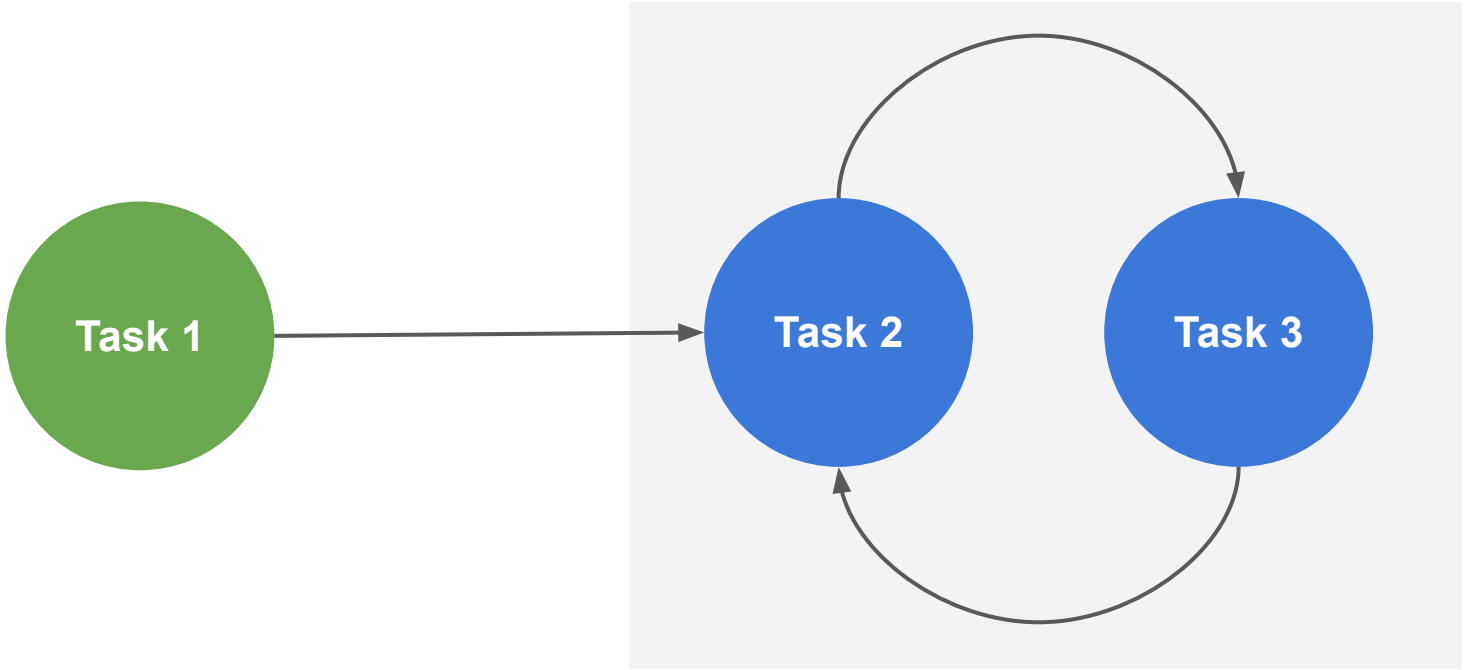
Directed Acyclic Graph



No dependencies.
These tasks can be started in any order

These tasks can only be started when all nodes connected to them have been completed.

Directed ~~A~~cyclic Graph



Task 2 and Task 3 are connected to each other in an infinite feedback loop. There is no logical termination point in this graph.

Outline

1. Communication
2. Motivation
3. What are Data Pipelines
4. Dask
5. Directed Acyclic Graph (DAGs)
6. **Computational Resources**
7. Task Scheduling
8. Cloud Storage

Computational Resources

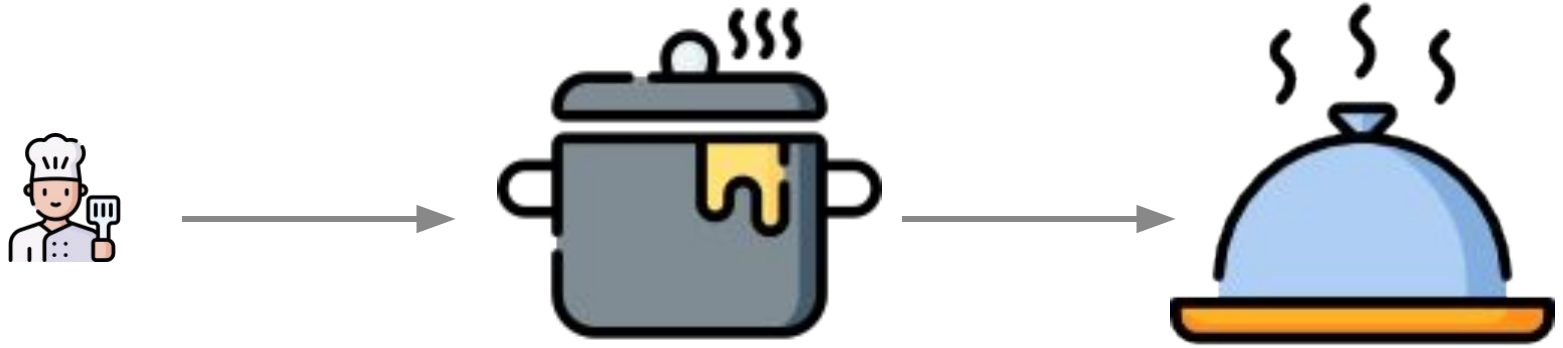
How to handle computational resources? As the problem we solve requires more resources we have two options:

- **scale up**: increase size of the available resource: invest in more efficient technology. **cons** diminishing return.

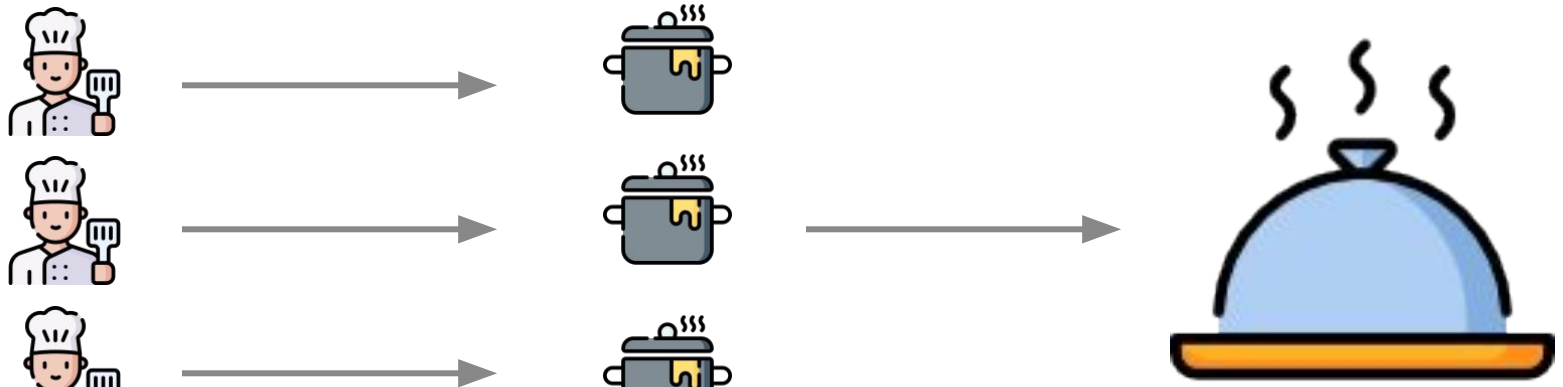
- **scale out**: add other resources (dask's main idea). Invest in more cheap resources. **cons** distribute workload.

Computational Resources

Scale up /
Vertical Scaling



Scale out /
Horizontal Scaling



Protopapas

Computational Resources

As we approach greater number of "work to be completed", some resources might be not fully exploited. This phenomenon is called **concurrency.**

For instance some might be idling because of insufficient shared resources (i.e. *resource starvation*). Schedulers handle this issue by making sure to provide enough resources to each worker.

Computational Resources

Dicing onions

Shared resources

Mincing garlic

This cook must wait and remain idle until either a knife becomes available or a new task that does not require a knife is available.
This is an example of a resource-starved worker.

Computational Resources

In case of a failure, *Dask* reaches a node and repeats the action without disturbing the rest of the process. There are two types of failures:

- **work failures**: a worker leaves, and you know that you must assign another one to their task. This might potentially slow down the execution, however it won't affect previous work (aka data loss).

- **data loss**: some accidents happens, and you have to start from the beginning. The scheduler stops and restarts from the beginning of the whole process.

Outline

1. Communication
2. Motivation
3. What are Data Pipelines
4. Dask
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. **Task Scheduling**
8. Cloud Storage

Task Scheduling

Dask performs a so called lazy computation. Until you run the method `.compute()`, *Dask* only splits the process into smaller logical pieces.

Even though the process is defined, the number of resources assigned and the place where the result will be stored are **not assigned** because the scheduler assigns them dynamically. This allow to recover from worker failure.

Task Scheduling

Dask uses a central scheduler to orchestrate the work.

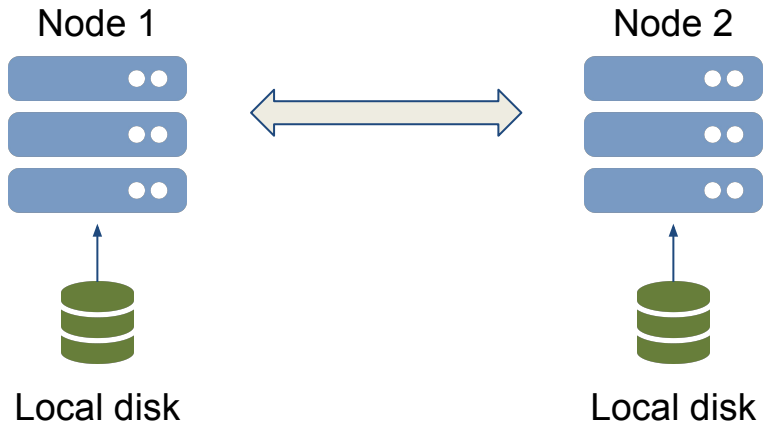
It splits the workload among different servers, which will unlikely be perfectly balanced with respect to load, power, and data access.

Due to these conditions, the scheduler needs to react to avoid bottlenecks that promptly affect overall runtime.

Task Scheduling

Assuming there are two nodes for computation.

Data needs to be replicated for each node in order to perform computation across nodes.

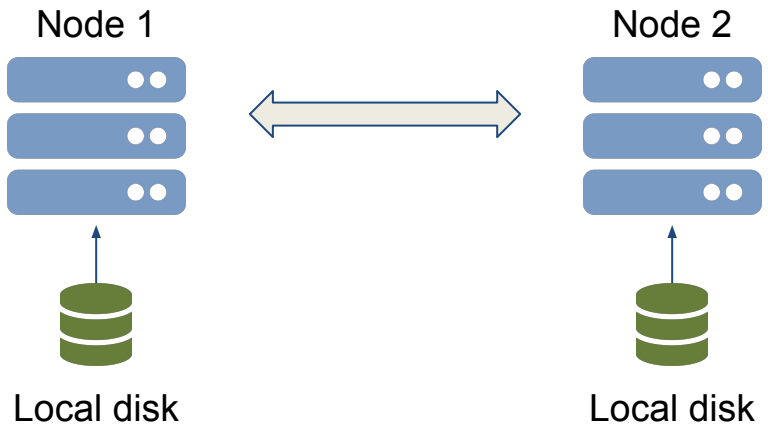


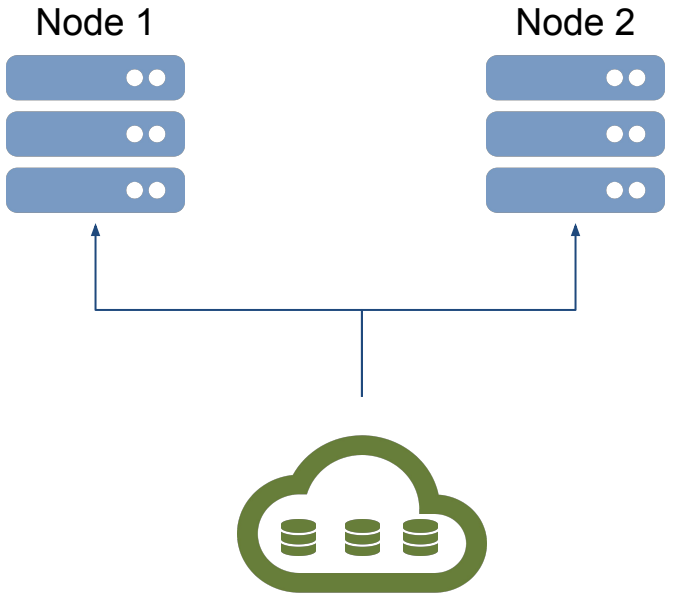
Task Scheduling

Assuming there are two nodes for computation.

Data needs to be replicated for each node in order to perform computation across nodes.

The remedy is to split data minimizing the number of data to broadcast across different local nodes.



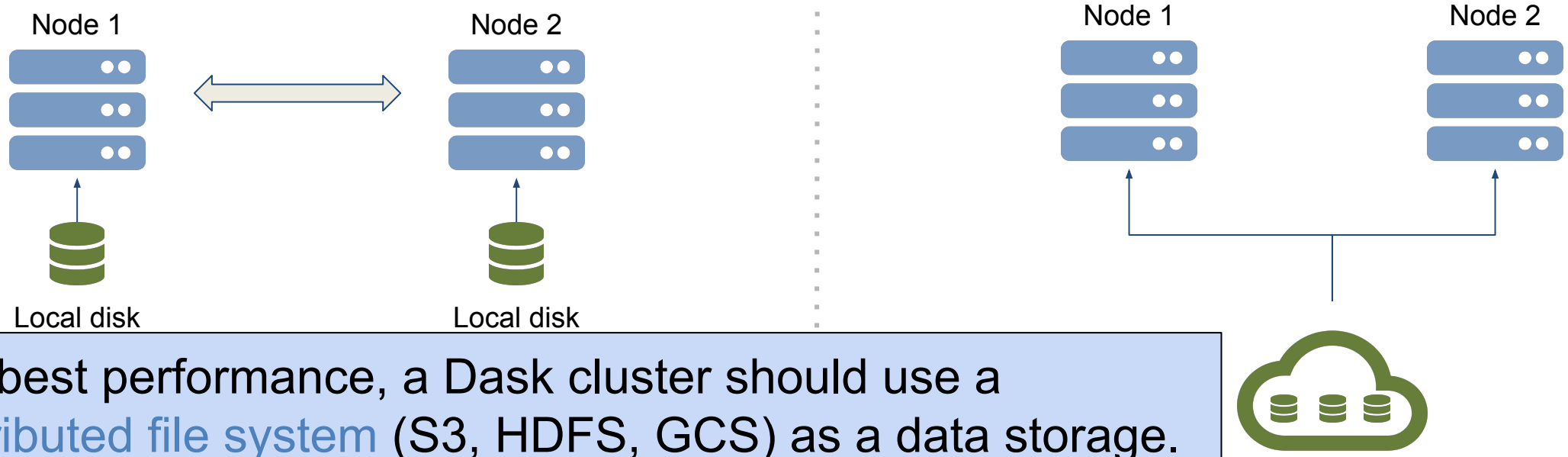


Task Scheduling

Assuming there are two nodes for computation.

Data needs to be replicated for each node in order to perform computation across nodes.

The remedy is to split data minimizing the number of data to broadcast across different local nodes.



For best performance, a Dask cluster should use a distributed file system (S3, HDFS, GCS) as a data storage.

Dask Review

- Dask can be used to scale popular Python libraries such as Pandas and NumPy allowing **to analyze dataset with greater size** (>8GB).
- Dask uses **directed acyclic graph to coordinate execution** of parallelized code across processors.
- Upstream actions are completed before downstream nodes.
- **Scaling out** (i.e. add workers) can improve performances of complex workloads, however, create overhead that can reduces gains.
- In case of failure, the step to reach a **node can be repeated** from the beginning without disturbing the rest of the process.

Dask Limitations

- Dask dataframe are immutable. Functions such as `pop` and `insert` are not supported.
- Dask does not allow for functions with a lot of data shuffling like `stack/unstack` and `melt`.
  - Do major filter and preprocessing in Dask and then dump the final dataset into Pandas.
- `Join, merge, groupby`, and `rolling` are supported but expensive due to shuffling.
  - Do major filter and preprocessing in Dask and then dump the final dataset into Pandas or limit operations only on index which can be pre-sorted.

Outline

1. Communication
2. Motivation
3. What are Data Pipelines
4. Dask
5. Directed Acyclic Graph (DAGs)
6. Computational Resources
7. Task Scheduling
8. **Cloud Storage**

Cloud Storage

Advantage of Cloud Storage:

- Unlimited capacity (Scalability)
- 99.99% uptime
- Distributed storage
- Data security
- Low cost

Examples:

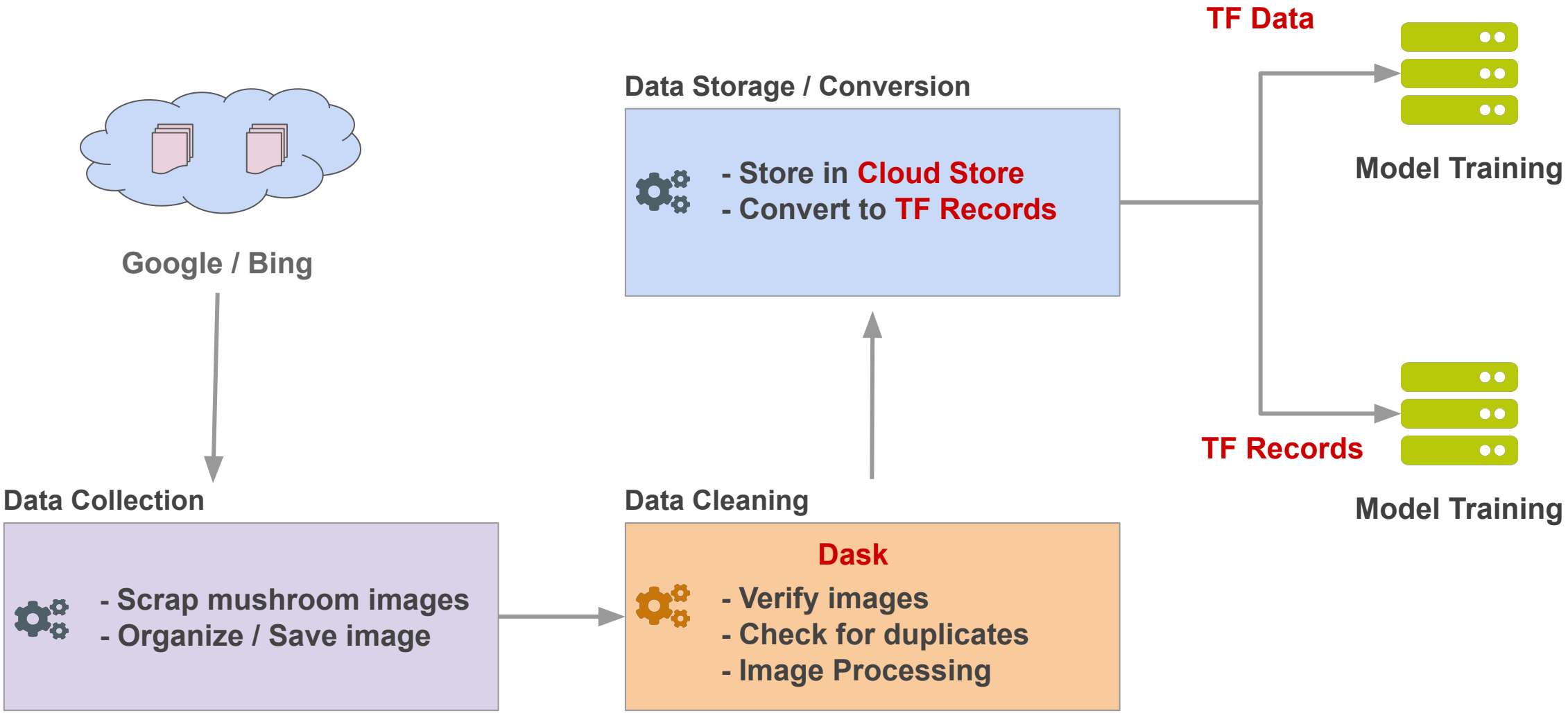
- AWS S3
- Azure data lake
- GCS (Google Cloud Storage)

Cloud Storage (GCS)

Google Cloud Storage:

- Object storage for companies of any size
- Store any amount of data and retrieve as often
- Reliable and secure object storage
- Multiple redundancy options (locations)
- Multiple storage classes
  - Standard
  - Nearline
  - Coldline
  - Archive

Putting it all together



THANK YOU