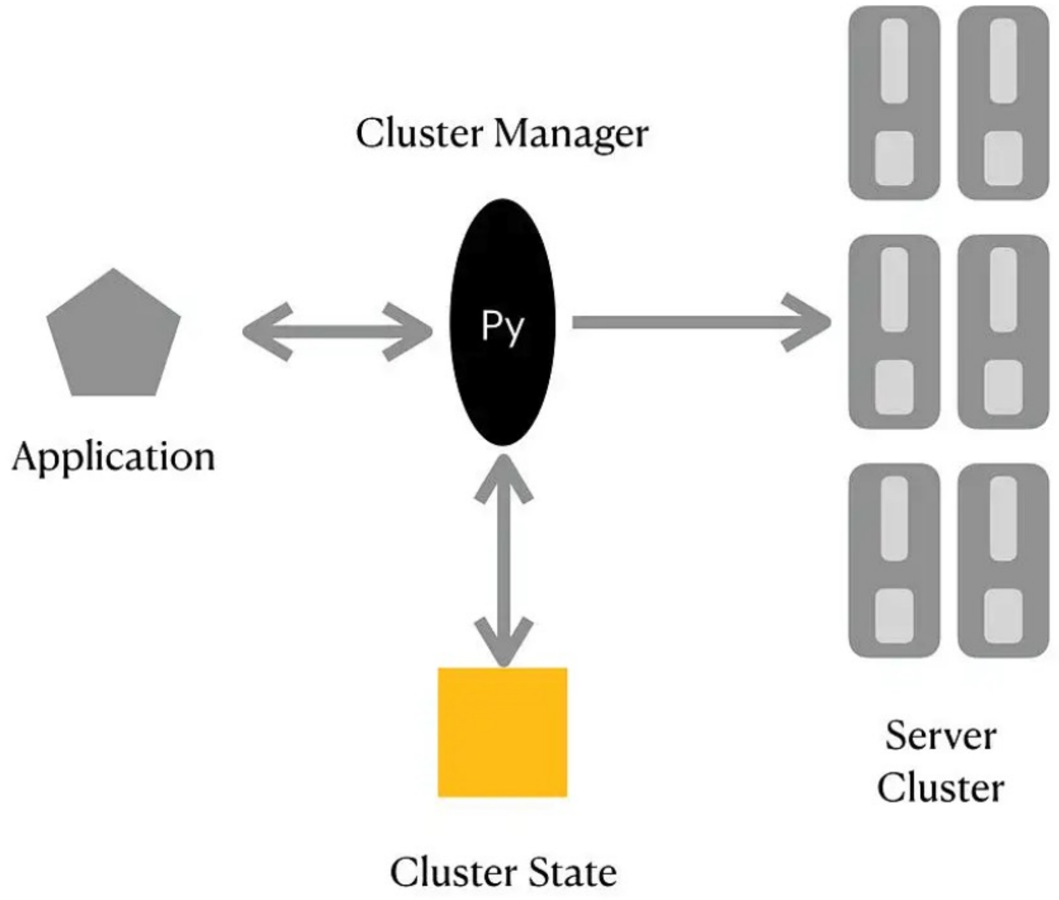
AI&Cloud Computing

TA - 4

A Simple Cluster

- You need to create cluster of machines in this project.
- The machines are docker containers
- Cluster manager is a Python program
- The state of the cluster is written to a file



Assignment #1: Create Cluster Manager

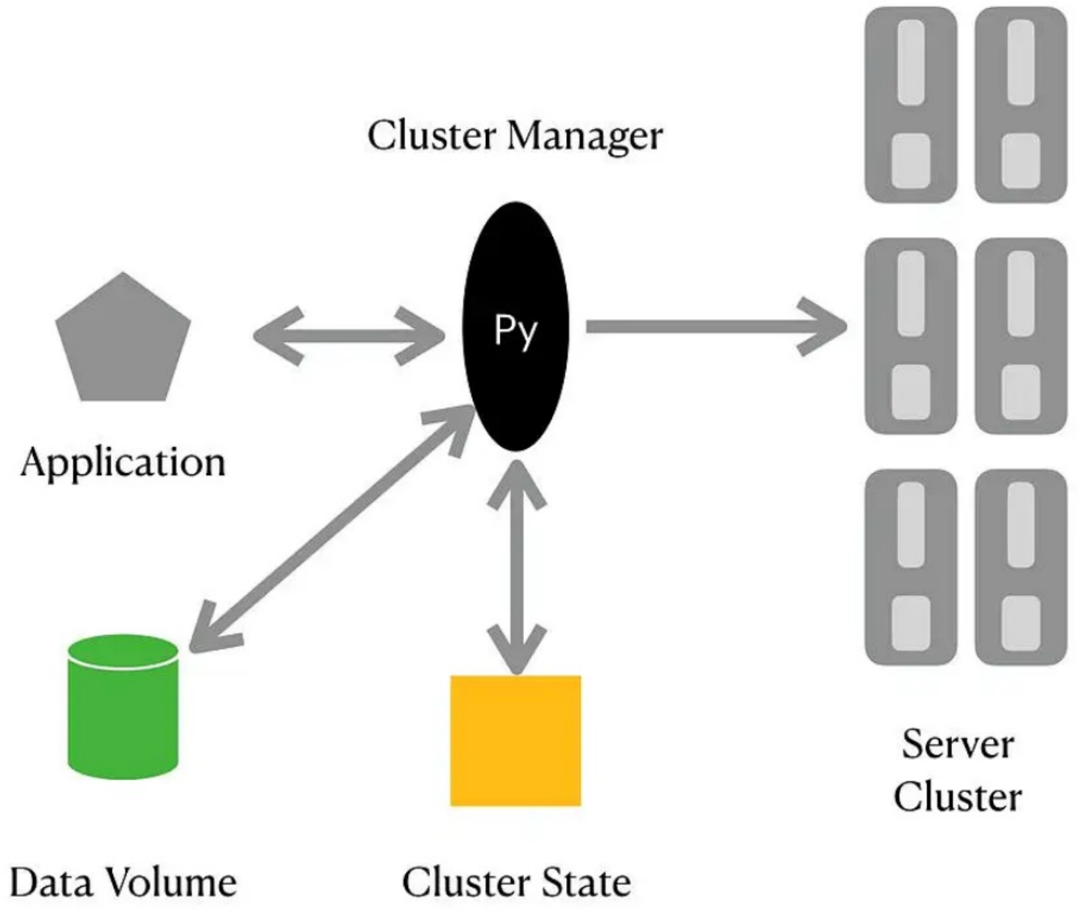
- Cluster manager (CM) is going to be a Python program
- You run the CM and it should be able to create the cluster of containers - let's say 8 containers are created by the CM
- You can think of the cluster as a simple cloud with 8 fake VMs
- In this assignment you need to support the following commands: create cluster, list cluster, run a simple command in the cluster, stop the cluster, delete all the containers in the cluster

Assignment #1: More Information

- Cluster create - you give the number of containers to start, the CM would create them - each container is a docker machine
- You can use docker tools - there is a Python interface to all the docker tools - so you can control docker containers using Python programs
- List the running containers - again just use the Python docker tools. To list, you ask the CM to list the containers and it does it for you
- Same way for other commands

Assignment #2: Data Processing in CM

- Container manager (CM) needs to support data volumes
- Data volume is a storage for data - the data you put there is available for all the containers
- Put a large array of data and it is available for all containers
- Each container can process a portion of the array
- They can print the local results to screen

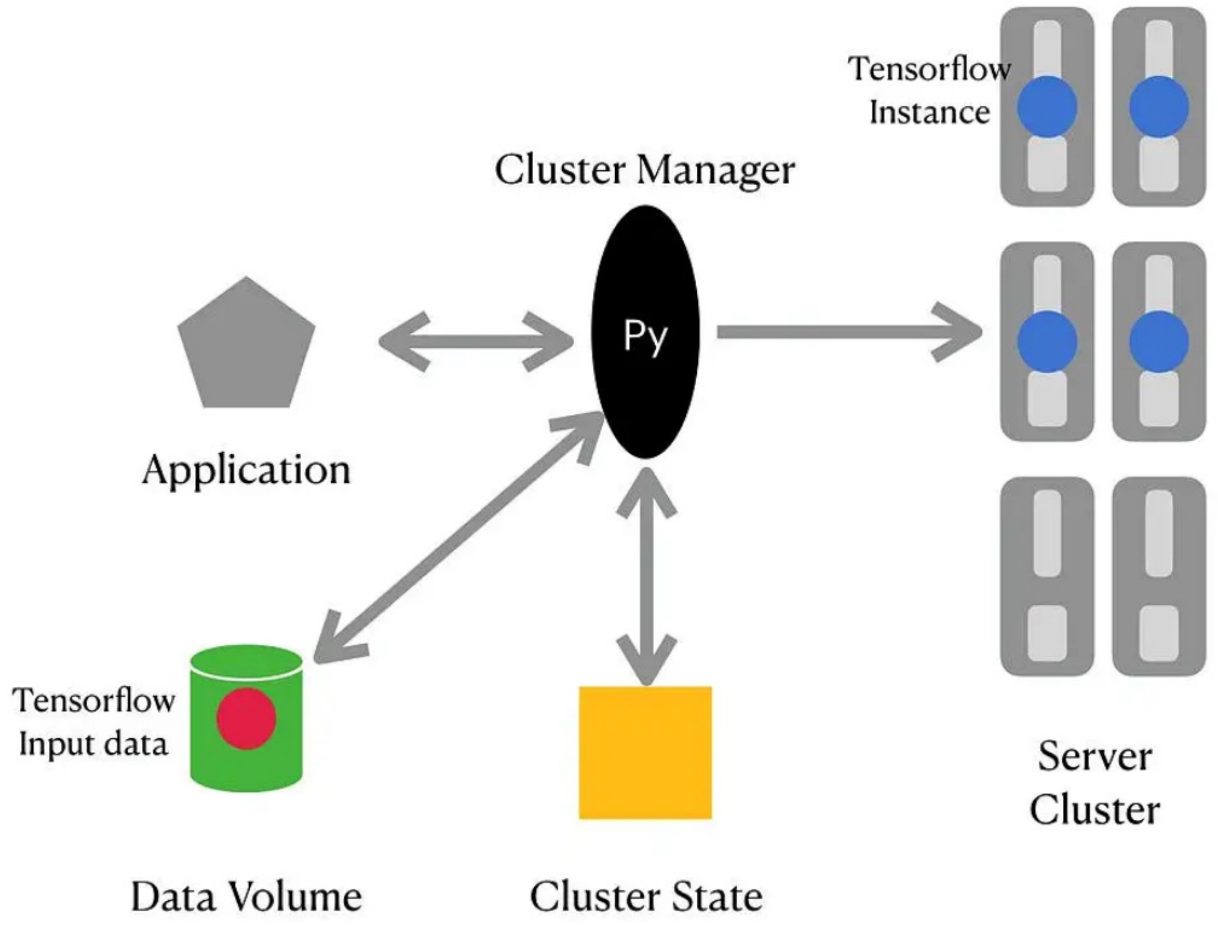


Assignment #2: Data Processing in CM

- Let's have array of 100,000 numbers
- We create a cluster with 4 containers
- Each container process 1/4 of the array - available through the data volume
- Each container prints a result for its portion - we ask each container to print the sum, average, max, min, standard deviation for their portions of the array
- So, we get four results for each parameter

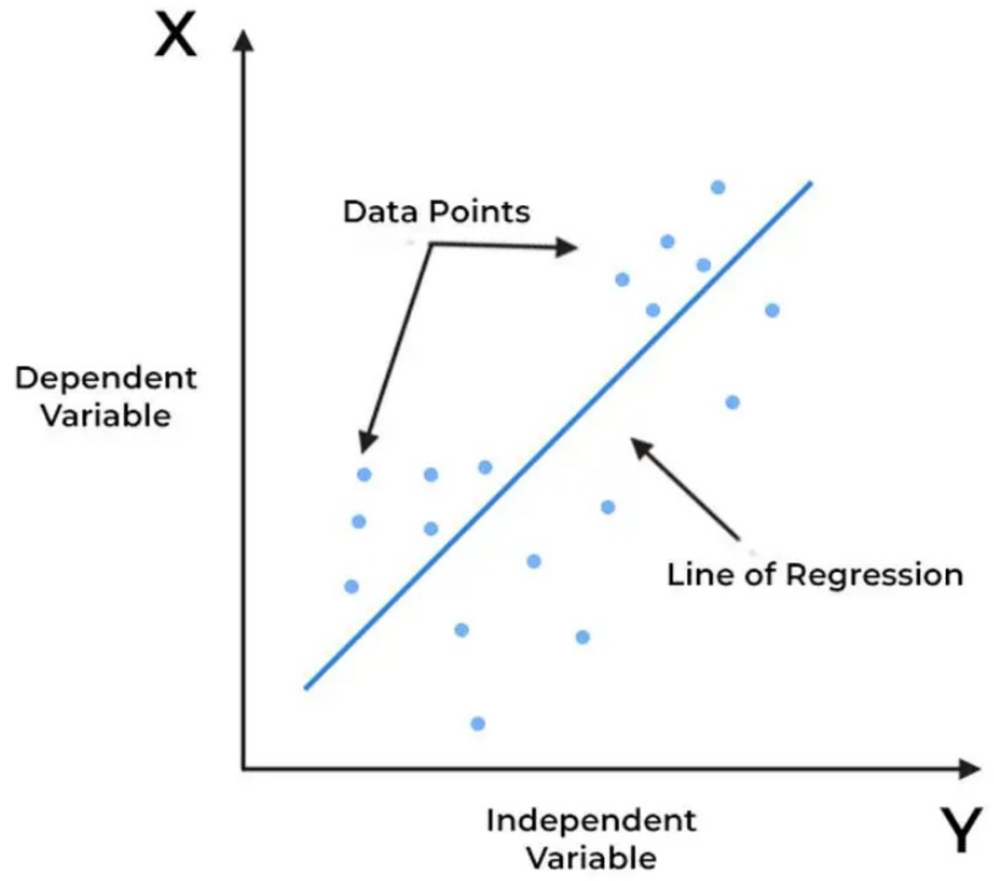
Assignment #3: AI Tasks in CM

- Cluster manager (CM) is dealing with Docker containers
- We can load Tensorflow into Docker containers quite easily
- Lets write a program to do linear regression on an example data set using Tensorflow



Linear Regression Using TensorFlow

```
import numpy as np
import tensorflow as tf
import matplotlib as plt

np.random.seed(555)
np.set_random_seed(555)

x = np.linspace(0, 60, 60)
y = np.linspace(0, 60, 60)

x += np.random.uniform(-5, 5, 60)
y += np.random.uniform(-5, 5, 60)

n = len(x)

plt.scatter(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Training data')
plt.show()
```



```
X = tf.placeholder("float")
Y = tf.placeholder("float")

W = tf.Variable(np.random.randn(), name = "W")
b = tf.Variable(np.random.randn(), name = "b")

learning_rate = 0.01
training_epochs = 1000

y_pred = tf.add(tf.multiply(X, W), b)
cost = tf.reduce_sum(tf.pow(y_pred - Y, 2) / (2 * n)
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
init = tf.global_variables_initializer()

with tf.Session() as tsess:
    tsess.run(init)
    for epoch in range(training_epochs):
        for (_x, _y) in zip(x, y):
            tsess.run(optimizer, feed_dict = {X: _x, Y: _y})
        if (epoch + 1) % 60 == 0:
            c = tsess.run(cost, feed_dict = {X: _x, Y: _y})
    training_cost = tsess.run(cost, feed_dict = {X: _x, Y: _y})
    weight = tsess.run(W)
    bias = tsess.run(b)

predictions = weight * x + bias
```

```
plt.plot(x, y, label= 'Original data'))
plt.plot('x', predictions, label = 'Fitted data')
plt.title('Linear Regression')
plt.legend()
plt.show()
```

How Evaluated?

- Assignment #1: 50 points - 10 points for each of the actions of CM
- Assignment #2: 30 points -
- Assignment #3: 20 points

ML: Linear Regression

https://en.wikipedia.org/wiki/Linear_regression

1. TensorFlow: https://www.tensorflow.org/

   Demo: https://www.geeksforgeeks.org/linear-regression-using-tensorflow/

2. PyTorch: https://pytorch.apachecn.org/#/

   Demo: https://www.geeksforgeeks.org/linear-regression-using-pytorch/