```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "df828baf-fa50-4a86-9432-1d85b1f78944",
   "metadata": {
    "tags": []
   },
   "source": [
    "# Assignment 5: Comparing Single Node vs Cluster Performance and Cost in Image Classification Using Azure Databricks\n",
    "\n",
    "# Description\n",
    "\n",
    "In this assignment, you will explore the impact of using a distributed system for running image classification. The aim is to gain an understanding of how a multi-node cluster varies from a single-node setup in terms of performance and cost. You will use Azure Databricks to run your experiments and the CIFAR-10 dataset for your image classification tasks. At the end of the assignment, you will be submitting the comparisons between the nodes and the result."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "f014eccf",
   "metadata": {},
   "source": [
    "### Prerequisite\n",
    "\n",
    "Cost tracking will be required in this assignment so please shutdown or terminate all old VMs / VNet / Azure Services that are active.\n",
    "\n",
    "We have predefined cost tags. Therefore the only cost tags that will work on your account are:\n",
    "- Tag: cpsc436:cost Value: scenario1\n",
    "- Tag: cpsc436:cost Value: scenario2\n",
    "\n",
    "For tracking costs, you can search tags in the top searchbar and filter the list of resources with the tag. You can also use the cost analysis tool in the Azure portal to track costs.\n",
    "\n",
    "For Azure resources, tags can be used while creating the Databricks cluster. For more information on how to use tags, please refer to the following link: \n",
    "\n",
```

```
"https://learn.microsoft.com/en-us/azure/databricks/administration-guide/account-settings/usage-detail-tags"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "4470f7d6-8b65-4041-aa66-df6630cea42a",
   "metadata": {},
   "source": [
    "## Learning Outcomes:\n",
    "After completing this assignment, you should be able to:\n",
    "- Gain experience working with Azure Databricks clusters.\n",
    "- Understand the basics of distributed machine learning.\n",
    "- Learn how to set up and deploy a machine learning model for image classification.\n",
    "- Explore the cost and performance implications of running different jobs on different numbers of nodes."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "686c2eda-6270-4201-a381-19e14cee1a1e",
   "metadata": {},
   "source": [
    "\n",
    "### Set Up an Azure Databricks Instance:\n",
    "Databricks stands out for its flexibility. It is designed to support a variety of distributed processing frameworks, such as Apache Spark and Hadoop, enabling users to run diverse data processing tasks. When it comes to specific applications like image classification, Databricks presents unique advantages. Image classification tasks, especially with deep learning models, usually require a significant amount of computational power and energy. Training models on large datasets can be computationally intensive and time-consuming. With Databricks' distributed capabilities, the dataset can be divided and processed concurrently across multiple nodes, drastically reducing the time taken for tasks like feature extraction and model training."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "1. Log in to the Azure portal.\n",
    "2. Navigate to the Databricks workspace page and click on \"+ Create Workspace\".\n",
```

```
    "3. Select existing workspace group or create a new one.\n",
    "4. For pricing tier, select Standard(Apache Spark)\n",
    "5. Specify the tags for scenario1 as mentioned in the
prerequisite.\n",
    "6. After databricks workspace is created, click on \"Launch
Workspace\".\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Scenario 1: Single Node"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "9f1d2ac4-6069-4d9c-8f78-faddac6fe2fa",
   "metadata": {},
   "source": [
    "In Scenario 1, we're first going to be looking at the analysis of a
single node. In this phase, we'll look into the configuration and
performance metrics of a single node to understand its capabilities and
limitations. By running our image classification tasks on a single node, we
can monitor its resource utilization, analyze the time taken for
processing, and evaluate the overall efficiency.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "1. Launch your Databricks workspace.\n",
    "2. Navigate to the Databricks workspace page and click on \"New\" and
select \"Cluster\n",
    "3. For this scenario, we will be working with single node
configuration. \n",
    "4. Use the latest Runtime version.\n",
    "5. For the node type, select Standard_DS3_v2. If your size is too
small, you will not be able to run pyspark.\n",
    "6. Make sure to set a time limit for the cluster termination to avoid
extra charges. You can set it to 1 hour.\n",
    "7. Click on create compute to create the cluster."
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "id": "b52a13dc-4ef2-4150-beda-e306af354318",
   "metadata": {},
   "source": [
    "#### Summary Info \n",
    "\n",
    "- Name: (anything, e.g. databricks-single-node)\n",
    "- Azure Databricks runtime: Latest available\n",
    "- Libraries: Spark\n",
    "- Cluster mode: Single-node\n",
    "- Worker type: Standard_DS3_v2\n",
    "- Cluster termination: Automatic (1-2 hour)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "806132fc-14de-4d7c-bac8-36a48c83e413",
   "metadata": {
    "tags": []
   },
   "source": [
    "### Upload CIFAR-10 Dataset to Azure Blob Storage"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "f6e7d13e-7092-4e64-b3cf-cba88a5bbb82",
   "metadata": {},
   "source": [
    "We will be working with the CIFAR-10 dataset. The CIFAR-10 dataset
contains 60,000 32x32 color images in 10 classes, with 6,000 images per
class. While there are several batches in the dataset, for simplicity,
we'll use just batch_1, which contains 10,000 images. We will be
downloading and uploading this dataset to an Azure Blob Storage container.
"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "e439cb2f-c6e8-4ab8-ba27-ac887ef6a3dd",
   "metadata": {
    "tags": []
   },
   "outputs": [],
   "source": [
```

```
    "# Download and extract the CIFAR-10 dataset\n",
    "!wget https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz\n",
    "!tar -xzf cifar-10-python.tar.gz"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "097c333b-c411-4c37-8024-1aa0c4411583",
   "metadata": {},
   "source": [
    "Since you are familiar with creating an Azure Blob Storage container (from Assignment 3), you will be uploading your image dataset to the Blob Storage container.\n",
    "\n",
    "1. Upload your image dataset (the entire folder) to this Blob Storage container.\n",
    "2. Ensure your Databricks workspace has permissions to access this container. Usually, the managed identity associated with your workspace has the right permissions to access the Blob Storage container. If not, you may need to assign the \"Storage Blob Data Contributor\" role to the managed identity. You can do this in the Azure portal, under the \"Access control (IAM)\" settings of the Blob Storage account. Remember to select the managed identity of your Databricks workspace when assigning the role. \n",
    "\n",
    "Note : To avoid this altogether, you can also assign the same resource group as your databricks workspace to your storage account.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "cbf0fc87-460b-42c4-808a-e81205488083",
   "metadata": {},
   "source": [
    "Once the upload is complete, you can verify the presence of your dataset files in the Blob Storage container using the Azure portal or Azure CLI."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "3dacdcf5-9b02-4a86-83c6-da177bb6cce6",
   "metadata": {},
   "source": [
    "### Perform Training on Dataset"
   ]
  },
```

```
  {
   "cell_type": "markdown",
   "id": "20544f54-aed8-45c5-8de4-54b6124fa92c",
   "metadata": {},
   "source": [
    "Since Databricks do not support SSH as of now, we can use databricks notebook to run our code. Lets start by creating a new notebook."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "bd539c12-05e3-4cba-bdf7-d1678761db78",
   "metadata": {},
   "source": [
    "Make sure to install these libraries on your databricks cluster to be able to process the dataset.\n",
    "- \"pip install tensorflow\" \n",
    "- \"pip install urllib3==1.26.6\""
   ]
  },
  {
   "cell_type": "markdown",
   "id": "20b8ab2e-65d5-4474-b2b3-ed705842deea",
   "metadata": {
    "tags": []
   },
   "source": [
    "First, we need to transfer the Blob Storage container contents of our data to our Databricks workspace. You can use the following code snippet to mount the storage container to your Databricks workspace.\n",
    "\n",
    "Make sure to replace the storage account name and container name with your own. Also choose a mount point for your files. Using this, we can locally call the files from the blob storage container.\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from pyspark import SparkConf, SparkContext\n",
    "from pyspark.sql import SparkSession\n",
    "\n",
    "# Define the connection to the Azure Blob storage\n",
```

```
    "container_name = \"<CONTAINER_NAME>\"\n",
    "storage_account_name = \"<STORAGE_ACCOUNT_NAME>\"\n",
    "sas_token = \"<SAS_TOKEN>\"\n",
    "\n",
    "# Define the mount point\n",
    "mount_point = \"/mnt/assignment5\"\n",
    "\n",
    "# Mount the Blob Storage\n",
    "dbutils.fs.mount(\n",
    "    source =
f\"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net\",
\n",
    "    mount_point = mount_point,\n",
    "    extra_configs =
{f\"fs.azure.sas.{container_name}.{storage_account_name}.blob.core.windows.
net\": sas_token}\n",
    ")"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "d9335702-1480-44e7-ae0b-6d158c02f09c",
   "metadata": {},
   "source": [
    "After installing the libraries, save the code below to a file called
\"train_cifar10.py,\" then transfer it to your instance using the scp
command. Once you're ready you can run \"python train_cifar10.py\" to begin
the preprocessing and training. "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "364867dc-c299-48a5-b3cd-37d2baa9a297",
   "metadata": {
    "tags": []
   },
   "outputs": [],
   "source": [
    "import tensorflow as tf\n",
    "import numpy as np\n",
    "import pickle\n",
    "import os\n",
    "\n",
    "def load_cifar10_batch(batch_filename):\n",
    "    \"\"\" Load a single batch from CIFAR10 \"\"\"\n",
```

```
    "    with open(batch_filename, 'rb') as file:\n",
    "        batch = pickle.load(file, encoding='latin1')\n",
    "        images = batch['data'].reshape((len(batch['data']), 3, 32,
32)).transpose(0, 2, 3, 1)\n",
    "        labels = batch['labels']\n",
    "        return images, labels\n",
    "\n",
    "def load_data(mounted_dir):\n",
    "    \"\"\" Load all CIFAR10 batches from the mounted directory
\"\"\"\n",
    "    images = []\n",
    "    labels = []\n",
    "    for i in range(1, 6):\n",
    "        batch_filename = os.path.join(mounted_dir,
f'data_batch_{i}')\n",
    "        batch_images, batch_labels =
load_cifar10_batch(batch_filename)\n",
    "        images.append(batch_images)\n",
    "        labels.append(batch_labels)\n",
    "    images = np.concatenate(images)\n",
    "    labels = np.concatenate(labels)\n",
    "    return images, labels\n",
    "\n",
    "def preprocess_images(images):\n",
    "    \"\"\" Preprocess images \"\"\"\n",
    "    images = images.astype(\"float32\") / 255  # Normalize to [0,
1]\n",
    "    return images\n",
    "\n",
    "def main():\n",
    "    mounted_dir = '/dbfs/'+mount_point\n",
    "    train_images, train_labels = load_data(mounted_dir)\n",
    "    train_images = preprocess_images(train_images)\n",
    "    train_labels = np.array(train_labels)\n",
    "\n",
    "    # Define the model\n",
    "    model = tf.keras.Sequential([\n",
    "        tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)),\n",
    "        tf.keras.layers.MaxPooling2D((2, 2)),\n",
    "        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),\n",
    "        tf.keras.layers.MaxPooling2D((2, 2)),\n",
    "        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),\n",
    "        tf.keras.layers.Flatten(),\n",
    "        tf.keras.layers.Dense(64, activation='relu'),\n",
    "        tf.keras.layers.Dense(10)\n",
```

```
    "    ])\n",
    "\n",
    "    # Compile the model\n",
    "    model.compile(optimizer='adam',\n",
    "
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),\n",
    "                  metrics=['accuracy'])\n",
    "\n",
    "    # Train the model\n",
    "    model.fit(train_images, train_labels, epochs=10)\n",
    "\n",
    "    # Load and evaluate test data\n",
    "    test_images, test_labels =
load_cifar10_batch(os.path.join(mounted_dir, 'test_batch'))\n",
    "    test_images = preprocess_images(test_images)\n",
    "    test_labels = np.array(test_labels)\n",
    "    test_loss, test_acc = model.evaluate(test_images, test_labels,
verbose=2)\n",
    "    print(f\"Test accuracy: {test_acc}\")\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    main()\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "6d080eb9-dfc9-4316-9075-17ac89c8515e",
   "metadata": {},
   "source": [
    "### Analysis and Comparison"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "5cffb9de-bf3c-4b9f-b7d5-697e7b270675",
   "metadata": {},
   "source": [
    "For the training time, document the training time for the single-node
cluster. Make sure to also note down any costs that are associated.\n",
    "Discuss factors that may affect the training times, e.g., data
distribution, model complexity, and the inherent parallelism of the
algorithm used.\n",
    "\n",
    "Once you complete these steps, you'll have successfully performed the
process on a single node and documented its performance and associated
costs. You can then proceed to Scenario 2 to repeat the process with a
```

```
5-node cluster. "
   ]
  },
  {
   "cell_type": "markdown",
   "id": "5b5f8721-fb0f-43b0-be3f-1bcb8f0c1ce3",
   "metadata": {
    "tags": []
   },
   "source": [
    "## Scenario 2: 5-Node Cluster Inference"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "c902ad89-f90b-4e02-ad6b-b88a117038f6",
   "metadata": {},
   "source": [
    "In Scenario 2, we are going to be recreating another Azure Databricks workspace and execute the same image classification inference job on this 5-node cluster. We want to then document the cost and time it takes to process the entire dataset on the 5-node cluster. One thing to note is that when running on a 5-node cluster, the data is automatically divided among the 5 nodes. This is managed by Databricks to facilitate distributed processing. Hence, you should expect the entire dataset to be processed faster.\n",
    "\n",
    "#### Important Note: \n",
    "In case you want to track the cost of your resources for scenario 2, you can create a new databricks cluster and use the tag for scenario2 as mentioned in the prerequisite. However to avoid repetition, the following steps will assume the same cluster used in scenario 1 with a tweak in the number of nodes.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "52536ea9-0f10-434c-a362-345a5ca931bf",
   "metadata": {},
   "source": [
    "### Set Up an Azure Databricks Workspace:\n",
    "1. Instead of single node in the databricks workspace, choose multi node.\n",
    "2. Keep everything similar to the single node configuration except for the number of worker nodes. \n",
    "3. For this scenario, we will be working with 5 worker nodes. By
```

```
default the auto scaling is enabled and can autoscale from a minimum of 2 nodes to a maximum of 8 nodes.\n",
    "4. Disable auto scaling and set the number of worker nodes to 5.\n",
    "5. Create the cluster"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "7fea06c8-05a3-4e94-94bb-2a983eb6bd0d",
   "metadata": {
    "tags": []
   },
   "source": [
    "In the right hand bar, there is a summary info. You can use this list below to cross-check and make sure that your settings are configured properly.\n",
    "\n",
    "#### Summary Info \n",
    "\n",
    "- Name: (anything, e.g. multi node)\n",
    "- Azure Databricks runtime: Latest available\n",
    "- Libraries: Spark\n",
    "- Cluster mode: Standard\n",
    "- Worker type: Standard_DS3_v2\n",
    "- Provisioning configuration: Head node: 1 instance, Worker nodes: 5 instances\n",
    "- Cluster termination: Automatically terminate cluster"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "6983f62c-9d9d-4cd1-b4b3-cfac0e703dfe",
   "metadata": {},
   "source": [
    "### Repeat the same steps for Training as scenario 1\n",
    "Using the 5-node cluster that you have created, perform training using the file: \"multi_node.py\". (The same version is pasted below). Similar to Scenario 1, you need to ensure that your Blob Storage container is first connected to your Databricks workspace and make sure you have tensorflow, urllib3==1.26.6, and pyspark installed.\n",
    "\n",
    "In this section, we're utilizing spark's parallelize function to distribute the batch file processing across Databricks nodes. The model is defined and broadcasted to all nodes to ensure that each node has a copy of the model.\n",
    "Batch files are processed in parallel across nodes, and each node
```

```
trains the model independently on its assigned batch. You may need to
modify the path where your files are located on the Databricks workspace
(\"/mnt/assignment5\"). For each step, make sure you record the start and
end times to calculate the training time.\n",
    "\n",
    "1. Train the model on the multi-node cluster.\n",
    "2. Record the start and end times to calculate the training time."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "b5d7a885-9fa6-4143-94af-08d28a0e49be",
   "metadata": {},
   "outputs": [],
   "source": [
    "from pyspark.sql import SparkSession\n",
    "from pyspark import SparkContext\n",
    "import tensorflow as tf\n",
    "import numpy as np\n",
    "import pickle\n",
    "import os\n",
    "\n",
    "# Initialize Spark session\n",
    "spark = SparkSession.builder.appName(\"CIFAR-10
Processing\").getOrCreate()\n",
    "\n",
    "# Access the SparkContext from the SparkSession\n",
    "sc = SparkContext.getOrCreate()\n",
    "\n",
    "def load_cifar10_batch(batch_filename):\n",
    "    \"\"\" Load a single batch from CIFAR10 \"\"\"\n",
    "    with open(batch_filename, 'rb') as file:\n",
    "        batch = pickle.load(file, encoding='latin1')\n",
    "        images = batch['data'].reshape((len(batch['data']), 3, 32,
32)).transpose(0, 2, 3, 1)\n",
    "        labels = batch['labels']\n",
    "        return images, labels\n",
    "\n",
    "def preprocess_images(images):\n",
    "    \"\"\" Preprocess images \"\"\"\n",
    "    images = images.astype(\"float32\") / 255  # Normalize to [0,
1]\n",
    "    return images\n",
    "\n",
    "def train_model(images, labels):\n",
```

```
    "    # Initialize TensorFlow distributed strategy\n",
    "    strategy = tf.distribute.MirroredStrategy()\n",
    "\n",
    "    with strategy.scope():\n",
    "        # Define the model\n",
    "        model = tf.keras.Sequential([\n",
    "            tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)),\n",
    "            tf.keras.layers.MaxPooling2D((2, 2)),\n",
    "            tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),\n",
    "            tf.keras.layers.MaxPooling2D((2, 2)),\n",
    "            tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),\n",
    "            tf.keras.layers.Flatten(),\n",
    "            tf.keras.layers.Dense(64, activation='relu'),\n",
    "            tf.keras.layers.Dense(10)\n",
    "        ])\n",
    "\n",
    "        # Compile the model\n",
    "        model.compile(optimizer='adam',\n",
    "
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),\n",
    "                      metrics=['accuracy'])\n",
    "\n",
    "    # Convert to TensorFlow dataset\n",
    "    train_dataset = tf.data.Dataset.from_tensor_slices((images,
labels))\n",
    "    train_dataset = train_dataset.batch(32)  # You may adjust the
batch size\n",
    "\n",
    "    # Train the model\n",
    "    model.fit(train_dataset, epochs=10)\n",
    "\n",
    "    return model\n",
    "\n",
    "def main():\n",
    "    data_dir = '/home/hadoop'  # CIFAR-10 batch files location, may
need to update this\n",
    "    batch_files = [os.path.join(data_dir, f'data_batch_{i}') for i in
range(1, 6)]\n",
    "\n",
    "    # Load all CIFAR-10 batches using Spark\n",
    "    batch_data =
sc.parallelize(batch_files).map(load_cifar10_batch).collect()\n",
    "    images_list, labels_list = zip(*batch_data)\n",
    "    train_images = np.concatenate(images_list)\n",
    "    train_labels = np.concatenate(labels_list)\n",
```

```
    "\n",
    "    train_images = preprocess_images(train_images)\n",
    "    train_labels = np.array(train_labels)\n",
    "\n",
    "    # Train the model on the EMR cluster with distributed
TensorFlow\n",
    "    model = train_model(train_images, train_labels)\n",
    "\n",
    "    # Load and evaluate test data\n",
    "    test_images, test_labels =
load_cifar10_batch(os.path.join(data_dir, 'test_batch'))\n",
    "    test_images = preprocess_images(test_images)\n",
    "    test_labels = np.array(test_labels)\n",
    "\n",
    "    test_dataset = tf.data.Dataset.from_tensor_slices((test_images,
test_labels)).batch(32)\n",
    "    test_loss, test_acc = model.evaluate(test_dataset, verbose=2)\n",
    "    print(f\"Test accuracy: {test_acc}\")\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    main()\n",
    "\n",
    "# Stop the Spark session\n",
    "spark.stop()"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "aa971ee4-9a5b-4975-9ec8-7e018e548588",
   "metadata": {},
   "source": [
    "### Analysis and Comparison"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "65c20d5b-d7a0-4eae-8d10-0b0bc2d251cd",
   "metadata": {},
   "source": [
    "In this next step, we want to perform some performance
measurement.\n",
    "\n",
    "For Document Processing Time:\n",
    "1. Compare the training times between the single node and multi-node
cluster.\n",
    "\n",
```

```
    "For Cost Tracking:\n",
    "1. Visit the Azure Cost Management + Billing in the Azure portal.\n",
    "2. Filter by the Databricks service and date range corresponding to your job execution.\n",
    "3. Document the cost associated with the Databricks usage for this multi-node run."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "1b5d3430-f1e1-43fa-b70b-23b99135ca37",
   "metadata": {},
   "source": [
    "##  Clean Up"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "79a810a6-c8db-494e-8641-7fa1c789a497",
   "metadata": {},
   "source": [
    "#### Terminate the Cluster:\n",
    "Once you are finished, you will eventually want to terminate the cluster. You can do this either from the Azure portal or by using the Azure Command Line Interface (CLI):\n",
    "```\n",
    "az databricks workspace delete --name your-workspace-name --resource-group your-resource-group-name\n",
    "```\n",
    "replace your-workspace-name with the actual name of your Databricks workspace and your-resource-group-name with the name of the resource group where your workspace is located. Please note that this command will delete the entire workspace, not just the cluster. If you only want to terminate the cluster, you can do so from the Databricks workspace UI."
   ]
  },
  {
   "cell_type": "markdown",
   "id": "03fdf727-ac92-4a88-9232-e8f3342e5782",
   "metadata": {},
   "source": [
    "## Final Analysis & Conclusion"
   ]
  },
  {
   "cell_type": "markdown",
```

```
   "id": "eaf7b32e-a671-4f45-9230-ffb8bece8723",
   "metadata": {},
   "source": [
    "After going through both scenarios, you can compare the results. Since the CIFAR-10 dataset is relatively small, the 5-node cluster doesn't speed up processing notably as the overhead of the cluster is compensated by data parallelism. As the complexity of the tasks increases, however, the advantages of parallel processing become more apparent. For larger datasets or more computationally intensive algorithms, the cluster's ability to distribute the workload across multiple nodes can significantly reduce processing time. \n",
    "\n",
    "This is particularly true for deep learning models, where the increase in computational resources can dramatically improve performance and training times. The main takeaway is that the effectiveness of a cluster in speeding up processing is highly dependent on the size and complexity of the dataset and tasks. In scenarios where data is very large and tasks are computationally demanding, a multi-node cluster can provide substantial benefits in terms of speed and efficiency.\n",
    "\n",
    "In terms of cost, operating a multi-node cluster is typically higher due to the increased hardware requirements, energy consumption, and potential need for different types of maintenance and management. This is especially relevant when dealing with cloud-based services, since the biggest cost factors would be the level of computing resources consumed, such as CPU/GPU usage, memory, and storage. Therefore, while a cluster may offer performance advantages, these need to be weighed against the higher operational costs.\n",
    "#### Compare Performance: \n",
    "Based on the results, compare the time taken by the single node and the 5-node cluster to process the entire dataset along with the costs associated with running the the job on both setups. \n",
    "\n",
    "#### Reflection: Consider the following:\n",
    "- How significant was the speedup when using the 5-node cluster?\n",
    "- Does the performance improvement justify the additional cost?\n",
    "- Would using an even larger dataset for this job yield more pronounced differences in performance?"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "717a2ee6-124c-4db4-b0b4-abec42b405e7",
   "metadata": {},
   "source": [
    "## Deliverables:\n",
```

```
    "\n",
    "- A Google document containing a ~200 words detailed report that includes the following:\n",
    "    - Comparison and results for both single-node and multi-node configurations\n",
    "    - Performance and cost metrics for both the single node and 5-node cluster setups.\n",
    "    - Your analysis and conclusions based on the results.\n",
    "    - Any challenges faced and how they were overcome as part of this assignment\n",
    "    - Screenshots or logs that validate your results.\n",
    "\n",
    "- A picture of your Azure Databricks dashboard with all your clusters terminated (clusters are expensive and we want to ensure that these are not running)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "fd743845-5032-49fb-bc53-c2e14258f540",
   "metadata": {},
   "source": [
    "## Congratulations!\n",
    "In this assignment, you learned how to work with an Azure Databricks Cluster, specifically how to execute a training job on a cluster of type single-node vs multi-node. By setting up and comparing the performance of single-node and multi-node clusters, you've gained practical insights into the intricacies of distributed data processing, understanding both its advantages and potential disadvantages."
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
```

```
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.5"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```