

```

{
  "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 Amazon EMR"
      ]
    },
    {
      "cell_type": "markdown",
      "id": "da957cea",
      "metadata": {},
      "source": [
        "# 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 Amazon EMR 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": "dfa0f123",
      "metadata": {},
      "source": [
        "### Prerequisite\n",
        "Cost tracking will be required in this assignment so please shutdown  
or terminate all old EC2 / VPC / AWS 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",
        "Watch the following video to see how to track costs. Please note this  
video is for tracking costs on an EC2 instance. Tagging and Tracking costs  
on an EMR instance will slightly vary.\n"
      ]
    }
  ]
}

```

```

    "\n",
    "https://www.loom.com/share/3dfef5dae63f4b9ba17d363c5b091dcd?sid=77c96995-d
e51-4893-81e0-5b204d6616e4\n"
  ]
},
{
  "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 Amazon EMR 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": [
    "## Scenario 1: Single Node"
  ]
},
{
  "cell_type": "markdown",
  "id": "fe1b70bb-f962-48fd-a6e0-19e10eb8b56d",
  "metadata": {},
  "source": [
    "### Set Up an Amazon EMR Instance:"
  ]
},
{
  "cell_type": "markdown",
  "id": "14240f1f-b9a9-4923-88a5-2ef6cf30f2f6",
  "metadata": {},
  "source": [
    "EMR (Elastic MapReduce) 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, EMR 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 EMR's 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",
  "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."

```
]
},
```

```
{
  "cell_type": "markdown",
  "id": "2b7b8a66-8554-4fa5-9551-1f2c37bf3c3c",
  "metadata": {
    "tags": []
  },
```

```
"source": [
```

- "1. Log in to the AWS Management Console.\n",
- "2. Navigate to the Amazon EMR console and \"Create cluster\"\n",
- "3. Set the Amazon EMR Release to \"emr-6.15.0\"\n",
- "4. Set the Application Bundle to Spark Interactive, and the Operating System Options to \"Amazon Linux Release\"\n",
- "5. For cluster configuration, we are going to be working with a single-node configuration for this part of the assignment. Select m5.xlarge for the Primary node (also known as Master node). If your size is too small, you will not be able to run pyspark. We also do not need a core node or a task instance group in this case.\n",
- "6. Scroll down to \"Cluster termination and node replacement\" and set the cluster termination as \"Manually terminate cluster\"\n",
- "7. If you have not already created a key pair, you can scroll to Security configuration and EC2 key pair to create a new key pair. Move the key\_pair file to your working directory along with where this assignment is located.\n",
- "9. Finally, scroll to the bottom to the section Identity and Access

Management (IAM) roles and select \"Choose an existing service roles.\" Set your Service role and Instance Profile to the IAM role that you've created.  
\n\",

\" - Note: If you have not created an IAM Service Role or Instance Profile, follow the instructions below\n\",

\"10. For EC2 instance profile for Amazon EMR, select \"Create an instance profile\" and then select your instance profile (ideally the one you've created for your EMR) \n\",

\"11. Add Tags for tracking costs\n\",

\"11. Once you're ready, you can click \"Create Cluster\"\"

]

},

{

\"cell\_type\": \"markdown\",

\"id\": \"f9565ecc-1bf9-4981-abba-9077091fd1ac\",

\"metadata\": {},

\"source\": [

\"#### (Optional) For students who are stuck on \"no IAM role specified\" error:\n\",

\"\\n\",

\"1. Navigate to the AWS Management Console and go to the IAM service.\n\",

\"2. Select \"Roles\" from the left navigation menu and click the \"Create role\" button.\n\",

\"3. In the \"Select trusted entity\" section, choose \"AWS service\" as the trusted entity type.\n\",

\"4. Click on the dropdown for Use cases for other AWS Services and select \"EMR\".\n\",

\"5. Select the EMR option below, then click Next\n\",

\"6. Leave the permissions as default and click Next\n\",

\"7. Enter a unique value for the role name (e.g. EMR-Access-Role), then click \"Create Role\".\n\",

\"8. After creating a Service role, you can additionally create an instance profile for your EMR. In the same page as the roles, select \"Create Role.\" Follow the same steps as before but this time select \"EMR Role for EC2\" as your Use Case. You can set a meaningful role name (e.g. EMR-instance-profile). Then select Create role.\"

]

},

{

\"cell\_type\": \"markdown\",

\"id\": \"b52a13dc-4ef2-4150-beda-e306af354318\",

\"metadata\": {},

\"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. emr-single-node)\n",
  "- Amazon EMR release: emr-6.15.0\n",
  "- Application bundle: Spark (3.4.1)\n",
  "- Instance groups: Primary (p3.2xlarge)\n",
  "- Provisioning configuration: Core size: 1 instance, Task size: 1
instance\n",
  "- Cluster termination: Manually terminate cluster"
]
},
{
  "cell_type": "markdown",
  "id": "806132fc-14de-4d7c-bac8-36a48c83e413",
  "metadata": {
    "tags": []
  },
  "source": [
    "#### Upload CIFAR-10 Dataset to Amazon S3"
  ]
},
{
  "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 Amazon S3 bucket. "
  ]
},
{
  "cell_type": "markdown",
  "id": "22a8d6b4",
  "metadata": {},
  "source": [
    "- Go to the CIFAR-10 website and download the python version of the
dataset: https://www.cs.toronto.edu/~kriz/cifar.html"
  ]
},
{
  "cell_type": "markdown",

```

```

    "id": "097c333b-c411-4c37-8024-1aa0c4411583",
    "metadata": {},
    "source": [
      "Since you are familiar with creating an S3 bucket you will be
uploading your image dataset to the S3 bucket.\n",
      "\n",
      "1. Decompress the zip folder and upload your image dataset (the entire
folder) to this S3 bucket.\n",
      "2. Ensure your EMR role has permissions to access this bucket. Usually
\"EMR_EC2_DefaultRole\" associated with your cluster has the right
permissions to access the S3 bucket."
    ]
  },
  {
    "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 S3 bucket using the AWS Management Console or AWS
CLI."
    ]
  },
  {
    "cell_type": "markdown",
    "id": "dcf80862",
    "metadata": {},
    "source": [
      "### Verify You Have SSH Permissions\n",
      "\n",
      "It is critical that you can SSH permissions on your EMR instance. To
verify SSH permissions complete the following steps:\n",
      "1) Once your EMR has been created go your EMR's main page.\n",
      "2) Scroll to the Tab \"Network And Security\"\n",
      "3) Click on the expandable \"EC2 security groups (firewall)\" text\n",
      "4) Under the title \"Primary Node\" you will see \"EMR managed
security group\". Click on the your security group that will be a link
highlighted in blue below it\n",
      "5) Click Edit inbound rules\n",
      "6) Add the following inbound rule if you don't already have it: Type:
SSH, Port 22, IP Address: 0.0.0.0/0"
    ]
  },
  {
    "cell_type": "markdown",
    "id": "3dacdcf5-9b02-4a86-83c6-da177bb6ccef",

```

```

"metadata": {},
"source": [
  "### Perform Training on Dataset"
]
},
{
  "cell_type": "markdown",
  "id": "20544f54-aed8-45c5-8de4-54b6124fa92c",
  "metadata": {},
  "source": [
    "SSH into your EMR Cluster. If you run into a permission error please
    verify you have SSH Permissions as per the previous step. You can SSH by
    either using:\n",
    "- AWS Command Line Interface (CLI): SSHing into your instance would
    look something like this: \"ssh -i [key-pair]
    hadoop@[Replace-with-your-primary node public DNS]\".\n",
    "- Terminal on AWS Website: Go to main page of EMR instance and click
    on \"Connect to the Primary node using SSM\". Choose the Tab \"EC2 Instance
    Connect\". In the username field type \"hadoop\". Press connect."
  ]
},
{
  "cell_type": "markdown",
  "id": "20b8ab2e-65d5-4474-b2b3-ed705842deea",
  "metadata": {
    "tags": []
  },
  "source": [
    "First we need to transfer the S3 bucket contents of our data to our
    EMR. You can run this command: "
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "id": "eab9b716",
  "metadata": {
    "vscode": {
      "languageId": "shellscript"
    }
  },
  "outputs": [],
  "source": [
    "aws s3 cp s3://[bucket-name]/cifar-10-batches-py/ /home/hadoop
    --recursive\n",
    "# (replace bucket-name with the name of your S3 bucket where the data

```

```

is)\n",
  "# (Note: --recursive makes sure to copy the files inside the
cifar-10-batches-py file)"
]
},
{
  "cell_type": "markdown",
  "id": "de3301d2",
  "metadata": {},
  "source": [
    "Make sure to install these libraries on your vm to be able to process
the dataset."
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "id": "6d40eb91",
  "metadata": {
    "vscode": {
      "languageId": "shellscript"
    }
  },
  "outputs": [],
  "source": [
    "pip install tensorflow\n",
    "pip install urllib3==1.26.6"
  ]
},
{
  "cell_type": "markdown",
  "id": "d9335702-1480-44e7-ae0b-6d158c02f09c",
  "metadata": {},
  "source": [
    "After installing the libraries you need to create a python script on
your VM named \"train_cifar10.py\". You can do this by either:\n",
    "- Use commands such \"nano\" in your VM's terminal to create the
file\n",
    "- Create the file locally and transfer it to your VM using the scp
command. \n",
    "\n",
    "Once the file is on your VM you can run the following command to begin
the preprocessing and training:"
  ]
},
{

```

```

"cell_type": "code",
"execution_count": null,
"id": "ea5f6ae7",
"metadata": {
  "vscode": {
    "languageId": "shellscript"
  }
},
"outputs": [],
"source": [
  "# Run this on you VM\n",
  "python train_cifar10.py"
]
},
{
"cell_type": "code",
"execution_count": null,
"id": "364867dc-c299-48a5-b3cd-37d2baa9a297",
"metadata": {
  "tags": []
},
"outputs": [],
"source": [
  "# train_cifar10.py\n",
  "\n",
  "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,\n32)).transpose(0, 2, 3, 1)\n",
  "        labels = batch['labels']\n",
  "        return images, labels\n",
  "\n",
  "def load_data(data_dir):\n",
  "    \"\"\" Load all CIFAR10 batches \"\"\"\n",
  "    images = []\n",
  "    labels = []\n",
  "    for i in range(1, 6):\n",
  "        batch_filename = os.path.join(data_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",
    "    data_dir = '/home/hadoop' # CIFAR-10 batch files location\n",
    "    train_images, train_labels = load_data(data_dir)\n",
    "    train_images = preprocess_images(train_images)\n",
    "    train_labels = np.array(train_labels)\n",
    "\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(data_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()"
]
},
{
"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."
]
},
{
"cell_type": "markdown",
"id": "476b2ded-be4b-4a1d-85f0-561fc250dd0b",
"metadata": {},
"source": [
"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 EMR instance 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 EMR to facilitate distributed processing. Hence, you should
    expect the entire dataset to be processed faster."
  ]
},
{
  "cell_type": "markdown",
  "id": "52536ea9-0f10-434c-a362-345a5ca931bf",
  "metadata": {},
  "source": [
    "### Set Up an Amazon EMR Instance:"
  ]
},
{
  "cell_type": "markdown",
  "id": "a4944736-0236-405b-b1fa-aaac0e918a4e",
  "metadata": {},
  "source": [
    "1. Log in to the AWS Management Console.\n",
    "2. Navigate to the Amazon EMR console and \"Create cluster\"\n",
    "3. Set the Amazon EMR Release to \"emr-6.15.0\"\n",
    "4. Set the Application Bundle to Spark Interactive, and the Operating
    System Options to \"Amazon Linux Release\".\n",
    "5. For cluster configuration, we are going to be working with a
    multi-node configuration for this part of the assignment. Select r4.2xlarge
    for the Primary node, and then r4.2xlarge for the Core node. For this
    setup, we won't use Task nodes. However, if we wanted to expand the
    computational power of our cluster without adding more HDFS storage, we
    could add Task nodes.\n",
    "6. Scroll to where it says \"Provisioning configuration,\" and set the
    instance size of the core node to 4.\n",
    "7. Keep the cluster termination as \"Manually terminate cluster\",\n",
    "8. Select your EC2 Key pair (the same one you've created with your
  ]
}

```

```

single-node EMR)\n",
    "8. For IAM roles, select the same options as you did for the
single-node configuration for \"Choose an existing service role\" and
\"Choose an existing instance profile\". \n",
    "9. Add Tags for Tracking Costs\n",
    "10. Once you're ready, you can click \"Create 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 confugred
properly.\n",
        "\n",
        "#### Summary Info \n",
        "\n",
        "- Name: (anything, e.g. emr-multi-node)\n",
        "- Amazon EMR release: emr-6.15.0\n",
        "- Application bundle: Custom (Spark 3.4.1)\n",
        "- Instance groups: Primary (r4.xlarge), Core (r4.xlarge)\n",
        "- Provisioning confugration: Core size: 1 Primary, 4 instances\n",
        "- Cluster termination: Manually terminate cluster"
    ]
},
{
    "cell_type": "markdown",
    "id": "6983f62c-9d9d-4cd1-b4b3-cfac0e703dfe",
    "metadata": {},
    "source": [
        "### Repeat the same steps for Training as scenario 1"
    ]
},
{
    "cell_type": "markdown",
    "id": "3e260a11",
    "metadata": {},
    "source": [
        "\n",
        "In this section, we're utilizing spark's parallelize function to
distribute the batch file processing across EMR 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 EMR server
(\\\"/home/hadoop\\\"). For each step, make sure you record the start and end
times to calculate the training time. "
    ]
  },
  {
    "cell_type": "markdown",
    "id": "042af042-70ae-4841-a088-d4207d20d9bc",
    "metadata": {},
    "source": [
      "Using the 5-node cluster, that you have created, perform training
using the file: \\\"multi_node.py\\\\" (see below). \n",
      "\n",
      "1) Download CIFAR dataset from your S3 bucket to your VM with the same
command used previously\n",
      "\n",
      "2) Install these libraries on your VM so you can process the dataset."
    ]
  },
  {
    "cell_type": "code",
    "execution_count": null,
    "id": "ce39293a",
    "metadata": {
      "vscode": {
        "languageId": "shellscript"
      }
    },
    "source": [
      "pip install tensorflow\n",
      "pip install urllib3==1.26.6\n",
      "pip install pyspark #!!!!!!!!!!!!!! Don't forget to install pyspark"
    ]
  },
  {
    "cell_type": "markdown",
    "id": "436856f9-2b52-427c-9be5-1838d484fcea",
    "metadata": {},
    "source": [
      "3) Train the model on the multi-node cluster.\n",
      "\n",
      "4) 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": [
    "# multi_node.py\n",
    "\n",
    "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\n",
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,\n",
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,\n",
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 AWS Cost Explorer in the AWS Management Console.\n",
        "2. Filter by the EMR service and date range corresponding to your job
execution.\n",
        "3. Document the cost associated with the EMR 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:"
  ]
},
{
  "cell_type": "markdown",
  "id": "c6904b5c-728b-4ee0-ba25-9e60d545d50e",
  "metadata": {},
  "source": [
    "Once you are finished, you will eventually once to terminate the
cluster. You can do this either from the AWS Management Console or by using
the AWS Command Line Interface (CLI):"
  ]
},
{
  "cell_type": "markdown",
  "id": "48a7ff38-8bb4-4b90-9929-18513f7dded6",
  "metadata": {},
  "source": [
    "````\n",
    "aws emr terminate-clusters --cluster-ids your-cluster-id\n",
    "````\n",
    "replace your-cluster-id with the actual ID of your EMR cluster."
  ]
},
{
  "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."
  ]
},
{
  "cell_type": "markdown",
  "id": "04095a9b-8ec8-4bac-8bb1-fc558c5982e5",
  "metadata": {},
  "source": [
    "#### 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",
    "- A picture of your EMR 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 Amazon EMR
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
}
```