

Spark Performance Comparison on Cloud Report		
Doc # SPCCR	Version: 1.4	Page 1 / 10

# **Spark Performance Comparison on Cloud & Blockchain Marketplace deployment using Azure**

**Baris Ozbas**  
**Burak Deniz**  
**Cenker Karaors**

Spark Performance Comparison on Cloud Report		
Doc # SPCCR	Version: 1.4	Page 2 / 10

## **TABLE OF CONTENTS**

### **Revision History 1**

### **1 Introduction 3**

**1.1 Document overview 3**

**1.2 References 3**

**1.3 Preparation 3**

### **2 Tests 4**

**2.1 Hardware Specification 4**

**2.2 Sample Data 4**

**2.3 Algorithms 4**

### **3 Performance and Price Comparison 5**

### **4 Conclusion 7**

### **5 Extra Content 8**

Spark Performance Comparison on Cloud Report		
Doc # SPCCR	Version: 1.4	Page 3 / 10

## 1 Introduction

### 1.1 Document overview

This document is the report of the CS450 project. It contains a performance comparison of spark in different VM instances provided by different Cloud providers.

### 1.2 References

<https://spark.apache.org/>

<https://jupyter.org/>

<https://github.com/selva86/datasets/blob/master/BostonHousing.csv> [1]

<https://www.kaggle.com/sazid28/advertising.csv> [2]

### 1.3 Preparation

#### 1.3.1 Setup

First of all, we ensured that required modules are installed and everything is up to date by running the below commands on the terminal for each machine.

- sudo apt-get update
- sudo apt install python
- sudo apt install python3
- sudo apt install python-pip
- sudo apt install python3-pip
- sudo python3 -m pip install pyspark numpy pandas
- sudo python -m pip install pyspark numpy pandas

#### 1.3.2 Testing Environment

After setting up the environment in each device described in section 2.1, we tested spark with reading and summing contents of a file that contains 10 Million lines.

## Spark Performance Comparison on Cloud Report

Doc # SPCCR

Version: 1.4

Page 4 / 10

## 2 Tests

This section contains hardware specifications, test inputs, and algorithms used during test execution.

### 2.1 Hardware Specification

Instances and local machines used in this project are described below.

#### *Amazon EC2 (EU Frankfurt):*

Instance Type	vCPU	ECU	Memory (GiB)	Instance Storage	Linux/UNIX Usage (dollars per Hour)
t2.medium	2	Variable	4GiB	EBS only	0.0536
t2.xlarge	4	Variable	16GiB	EBS only	0.2144
t3a.large	2	Variable	8GiB	EBS only	0.0864
t3a.2xlarge	8	Variable	32GiB	EBS only	0.3456
m5ad.4xlarge	16	N/A	64GiB	2 X 300 NVMe SSD	1

#### *Microsoft Azure:*

Instance Type	vCPU	Memory (GiB)	Instance Storage	Linux/UNIX Usage (dollars per Hour)
D2s_v3	2	8	Premium SSD	0.094

#### *Google (Cloud Engine):*

Instance Type	vCPU	Memory (GiB)	Instance Storage (GB)	Linux/UNIX Usage (dollars per Hour)
n1-standard	2	7.5	10	0.067

#### *Local Machine:*

Lenovo Legion: Intel i7-8750H CPU 2.20-2.21GHz - 16.0 GB RAM - 1 TB + 128 GB SSD

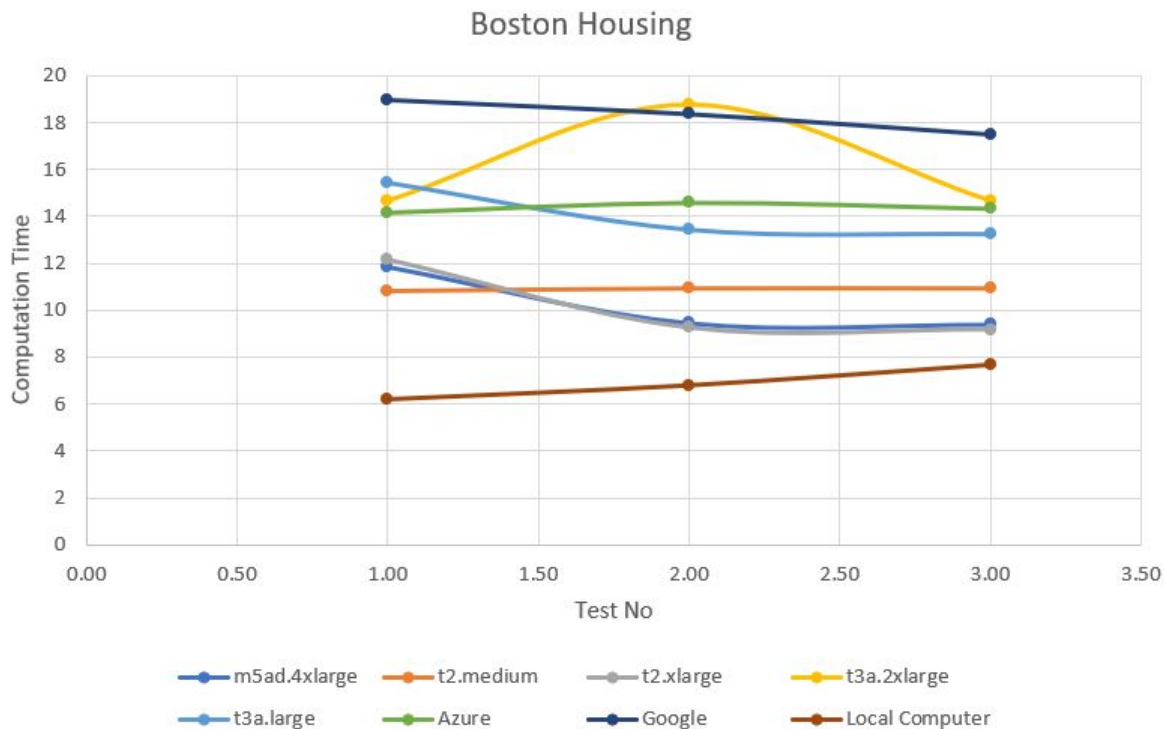
### 2.2 Sample Data

- 1) Boston Housing CSV [\[1\]](#)
- 2) Advertising CSV [\[2\]](#)

### 2.3 Algorithms

- The first algorithm that predicts the rows uses vector assembler and a linear regression algorithm. (boston.py)
- The second algorithm uses pipelines to fit data to the model to predict values. (advertisingv2.py)

### 3 Performance and Price Comparison



In our first algorithm, our aim is to do a prediction using linear regression and vector assembler on Boston housing prices. As you can see above our local computer has the fastest computation time when it comes to this algorithm. We believe that the reason for this is that our data set is not too big and can be cached easily by our local computer and we can train our models much faster because of this caching. The second-fastest machine to run this algorithm is m5ad.4xlarge which is expected because it has 16 virtual CPU and our most expensive machine. but you can see how close they are with t2.xlarge which cost only one-fifth of m5ad.4xlarge.

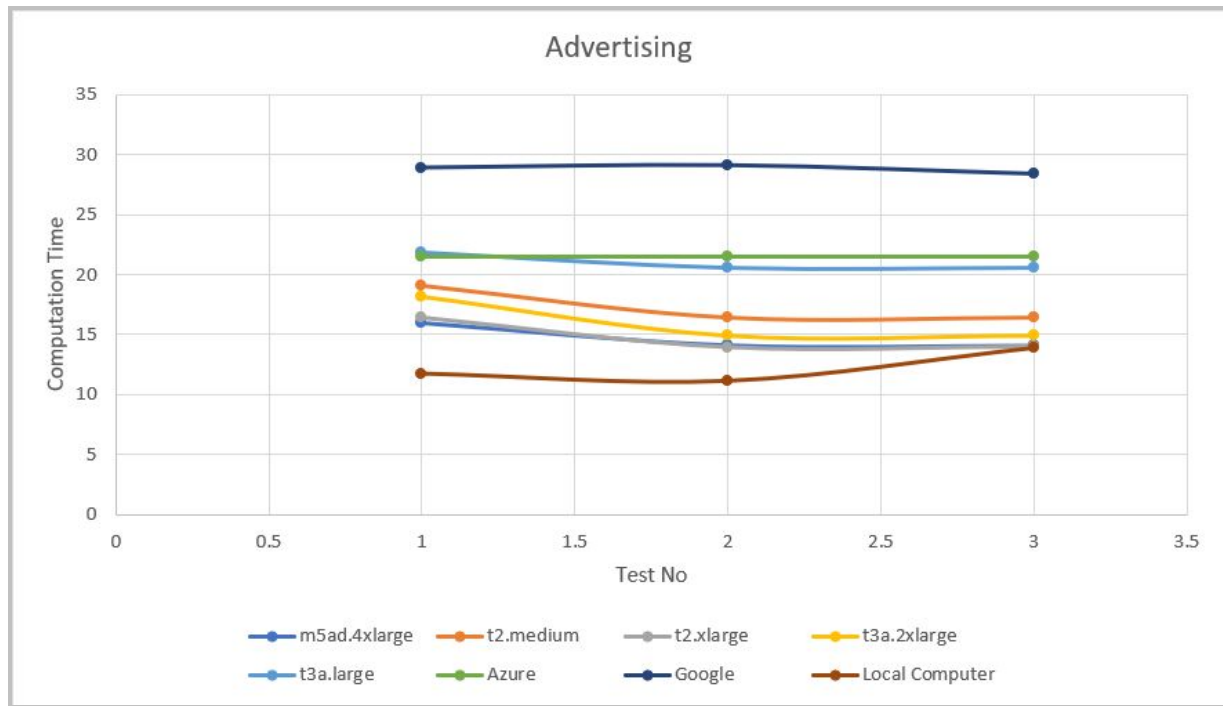
Also, we can see that Google has the slowest machines even though the specs of the devices don't differ much. From this comparison, we can deduct that in small data sets hardware capacity is not changing the computation time as much as expected. So, in conclusion, we can say that if a company wants to train an AI with a small data set the most productive and cost-efficient way of doing is getting a mediocre local machine or renting a virtual machine from Amazon's EC2 services.

## Spark Performance Comparison on Cloud Report

Doc # SPCCR

Version: 1.4

Page 6 / 10



In our second algorithm, we use pipelines to fit data to the model to predict values. Again our local computer has the fastest computation time because the data set is not very large and can be cached easily by our local machine. Surprisingly the Google has the slowest computation time in this algorithm and we can see that our local computer can run the same algorithm twice and still has 4 seconds before Google can finish the first run of the algorithm.

As you can see above that all t2.xlarge, m5ad.4xlarge, t3a.2xlarge, and t2.medium have almost the same computation time which brings us to the same conclusion which is in small data sets the hardware specs do not affect the computation time as much as expected and the best choice for a company would be to rent a virtual machine on Amazon or buy a mediocre laptop or desktop computer.

Spark Performance Comparison on Cloud Report		
Doc # SPCCR	Version: 1.4	Page 7 / 10

Instances	Boston Housing Average	Advertising Average	Price (dollars per Hour)
m5ad.4xlarge	10.2268	14.7555	1
t2.medium	10.8927	17.3227	0.0536
t2.xlarge	10.2081	14.8020	0.2144
t3a.2xlarge	16.0291	16.0077	0.3456
t3a.large	14.0357	20.9662	0.0864
D2s_v3	14.3523	21.5501	0.094
n1-standard	18.2749	28.8243	0.067
Local Computer	6.8956	12.2757	-

#### 4 Conclusion

In conclusion, we have identified that spark performs better with the small and medium-sized data sets, while the cloud environments perform better with larger data sets due to their elastic behavior. In terms of price cloud environments are more costly than local clusters, so we recommend local clusters for individuals and cloud for companies.

Spark Performance Comparison on Cloud Report		
Doc # SPCCR	Version: 1.4	Page 8 / 10

## 5 Extra Content: Marketplace Implementation and Deployment to Microsoft Azure Blockchain Services

### What is Microsoft Azure Blockchain Workbench?

Azure Blockchain Workbench Preview is a collection of Azure services and capabilities designed to help create and deploy blockchain applications to share business processes and data with other organizations. Azure Blockchain Workbench provides the infrastructure scaffolding for building blockchain applications enabling developers to focus on creating business logic and smart contracts. It also makes it easier to create blockchain applications by integrating several Azure services and capabilities to help automate common development tasks.

#### Key parts of Azure Blockchain Workbench:

- **Deploy a blockchain network:**

With an Azure Resource Manager solution template, Azure Blockchain Workbench simplifies consortium blockchain network setup as a preconfigured solution.

- **Use of Active Directory:**

Blockchain identities are represented as an address on the network using existing blockchain protocols. By associating it with an Active Directory identity, Azure Blockchain Workbench abstracts the blockchain identity, making it easier to build business applications with Active Directory identities.

- **Synchronize on-chain data with off-chain storage**

Azure Blockchain Workbench makes it easier to analyze blockchain events and data by automatically synchronizing blockchain data to storage off-chain. You can query off-chain database systems such as SQL Server instead of extracting data directly from the blockchain.

#### General Look and Deployment Process:

After successfully finishing the Microsoft Azure Blockchain Workbench setup and configuration, every each of service shown below will start working on the server.





















## Spark Performance Comparison on Cloud Report

Doc # SPCCR

Version: 1.4

Page 9 / 10

<input type="checkbox"/>  blockchain-dodutl	Application Insights
<input type="checkbox"/>  blockchain-dodutl	Key vault
<input type="checkbox"/>  blockchain-dodutl	App Service
<input type="checkbox"/>  blockchain-dodutl-api	App Service
<input type="checkbox"/>  blockchain-eg-dodutl	Event Grid Topic
<input type="checkbox"/>  blockchain-lb	Load balancer
<input type="checkbox"/>  blockchain-lb-public-ip	Public IP address
<input type="checkbox"/>  blockchain-plan	App Service plan
<input type="checkbox"/>  blockchain-sb-dodutl	Service Bus Namespace
<input type="checkbox"/>  blockchain-subnet-workers-nsg	Network security group
<input type="checkbox"/>  blockchain-vnet	Virtual network
<input type="checkbox"/>  blockchain-worker-n	Virtual machine scale set
<input type="checkbox"/>  blockchainedodutlbloc	Azure Blockchain Service
<input type="checkbox"/>  db-dodutl-blo	SQL server
<input type="checkbox"/>  dodutl-blo (db-dodutl-blo/dodutl-blo)	SQL database
<input type="checkbox"/>  dodutlblockchain	Storage account
<input type="checkbox"/>  website-api-test	Availability test
<input type="checkbox"/>  website-ui-test	Availability test

From load balancer to availability tester, even network security groups for blockchain workers are set up for easy testing and fast deployment of your smart contract on a blockchain. Also, the SQL server and database are working for the off-chain data storage as it can be seen and mentioned before.

For the deployment process, a solidity and a JSON file needs to be uploaded to the website UI.

Solidity is a high-level and contract-oriented language used for writing smart contracts. Developed by the core contributors of the Ethereum Blockchain Platform, it is used for designing and implementing smart contracts within the Ethereum Virtual Machine and other blockchain development platforms such as Microsoft Azure Blockchain. The JSON file is needed for Azure Blockchain Workbench configuration.

After uploading the required files to the system via website-UI service, the screen below will appear on the website.

## Spark Performance Comparison on Cloud Report

Doc # SPCCR

Version: 1.4

Page 10 / 10

Blockchain Workbench

Applications > Simple Marketplace

### Simple Marketplace

Workflow (version 1.0)

+ New | Customize table

Contracts

Id	State	Modified By	Modified	Description	Asking Price	Offer Price	Instance Owner
4	Item Available	Baris Ozbas	01/12/20	Taco	20	0	Baris Ozbas
3	Item Available	Burak Deniz	12/25/19	Kundura	100	250	Burak Deniz
2	Item Available	Baris Ozbas	12/25/19	1KG BAKLAVA	150	0	Baris Ozbas
1	Item Available	Baris Ozbas	12/25/19	Playstation 4	200	500	Baris Ozbas

On top right, members can be seen. Members that can be added to our marketplace are real Microsoft accounts which are in the *same organization* (we can add ozyegin or ozu domain accounts).

### Add a member

IA

Ismail Ari

×

Owner

Add

Cancel

There are 2 roles and 3 states on the application, the roles are Owner and Buyer. Owners can add item to be sold, and buyers can bid on the item. States used for items are ItemAvailable, OfferPlaced and Accepted.

An owner lists an item with 2 inputs, description and asking price. If a buyer bids on that item, the transaction state becomes OfferPlaced and starts to wait for the response from the actual owner.

If the owner accepts the offer, transaction state becomes Accepted and no more offer can be bid on the item, and it is considered as it is sold.