Sustainable and Scalable Setup for Teaching Big Data Computing

Linh B Ngo
West Chester University of Pennsylvania
West Chester, PA
lngo@wcupa.edu

Hoang Bui
Loyola University Maryland
Baltimore, MD
hdbui@loyola.edu

ABSTRACT

As more students want to pursue a career in big data analytics and data science, big data education has become a focal point in many colleges and universities' curricula. There are many challenges when it comes to teaching and learning big data in a classroom setting. One of the biggest challenges is to prepare big data infrastructure to provide meaningful hands-on experience to students. Setting up necessary distributed computing resource is a delicate act for instructors and system administrators because there is no one size fit all solutions. In this paper, we propose an approach that facilitates the creation of the computing environment on both personal computers and public cloud resources. This combined approach meet different needs and can be used in an educational setting to facilitate different big data learning activities. We discuss and reflect on our experience using these systems in teaching undergraduate and graduate courses.

KEYWORDS

Big Data Computing, Learning Activities, Apache Spark

1 INTRODUCTION

Multicore processors have become standard in modern personal computing devices. Linux Kernel Subsystem and Hypervisor components have ensured Windows-based computers to have access to the same software libraries commonly used in parallel and distributed computing (PDC) education such as *pthreads* [17], *OpenMP* [22] and *OpenMPI* [23]. This enables students to carry out PDC learning activities on their personal computers rather than fully dependent on large-scale computing resources. When it comes to big data computing (BDC) topics, computing environment setup becomes more complex. For example, Apache Spark, a popular big data analytic platform, is not a library to be linked and invoked at run time but a complex ecosystem that needs to be installed, configured, and deployed. Programs are then submitted to this platform for execution. In addition to multicore requirements, available memory and local storage are also critical resources to be managed. To date, BDC education relies mainly on distributed resources with a preference for on-site physical cluster [15].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Copyright ©JOCSE, a supported publication of the Shodor Education Foundation Inc.

© 2023 Journal of Computational Science Education
https://doi.org/10.22369/issn.2153-4136/14/1/7

In this work, we present several deployment varieties for individualized computing environments together with various BDC learning activities. These approaches range from direct installation and configuration on personal computing devices, development of workflow on local clusters to indirect deployment through containerization, and large-scale temporary deployment on federal cloud resources. This provides a sustainable approach to BDC education where the burden of maintain computing resources is not solely placed on academic institutions and students have access to a learning environment beyond the duration of the courses. These approaches help creating and disseminating BDC courses at two academic institutions that lack support for large-scale infrastructures.

The remainder of the paper is structured as follows. Section 2 presents our approach to maintain a sustainable BDC learning environment. Section 3 describes the learning activities and assessments. In Section 4 we discuss our overall classroom experience, including descriptions of previous taught courses, students evaluation and learning outcomes, and the lessons learned. Finally, we conclude our paper and discuss future work in Section 5.

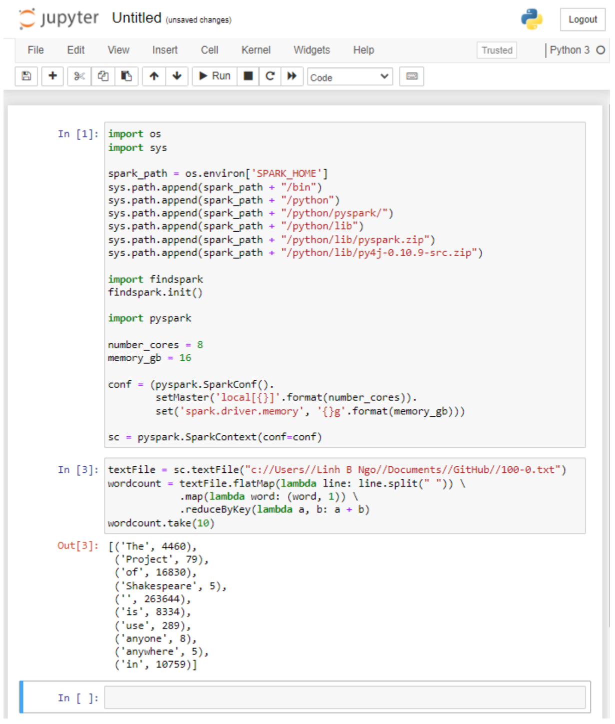
2 SUSTAINABLE AND SCALABLE SOLUTIONS

There are primarily three approaches to providing computing environment to big data education: physical cluster [7], virtual cluster [14], or cloud-based solutions [24]. Given these approaches require institutional investments and extensive technical knowledge, scalability and sustainability can be limited for smaller institutions. These solutions can become limited available to students due to limitation such as computing credits (cloud), on-campus access (physical or virtual), or resource contention (physical or virtual). Students can lose access to resources after the course is ended, hindering the potential of further self studying.

We define sustainable solutions as approaches that do not place significant financial and technical burden on students and academic institutions. A sustainable option for BDC learning environment, therefore, is one that is deployed on students' personal computer (PC). However, it is critical that learning activities behave exactly the same on personal computers or large-scale resources, except for run time performance.

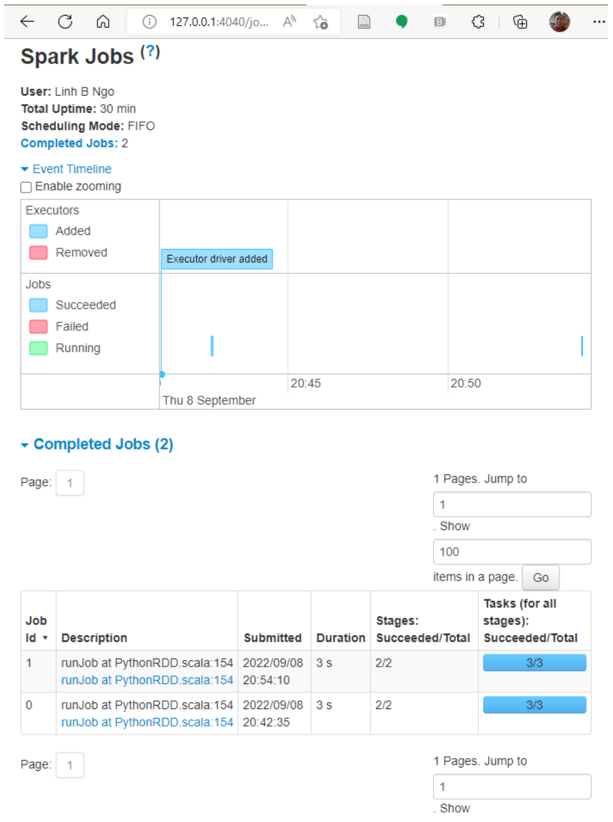
2.1 Infrastructure on personal computers

There are three varieties of deploying Spark on PC: direct deployment, single-node containerized local deployment, and multi-node containerized cluster deployment. Today, a fairly minimal and inexpensive (relative) laptop boasts a dual-core CPU, 4GB of memory, and 32GB of storage. A direct installation of Apache Spark [28]


jupyter Untitled (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]: import os
        import sys

        spark_path = os.environ['SPARK_HOME']
        sys.path.append(spark_path + "/bin")
        sys.path.append(spark_path + "/python")
        sys.path.append(spark_path + "/python/pyspark/")
        sys.path.append(spark_path + "/python/lib")
        sys.path.append(spark_path + "/python/lib/pyspark.zip")
        sys.path.append(spark_path + "/python/lib/py4j-0.10.9-src.zip")

        import findspark
        findspark.init()

        import pyspark

        number_cores = 8
        memory_gb = 16

        conf = (pyspark.SparkConf().
                setMaster('local[{}]'.format(number_cores)).
                set('spark.driver.memory', '{}g'.format(memory_gb)))

        sc = pyspark.SparkContext(conf=conf)

In [3]: textFile = sc.textFile("c://Users//Linh B Ngo//Documents//GitHub//100-0.txt")
        wordcount = textFile.flatMap(lambda line: line.split(" ")) \
                    .map(lambda word: (word, 1)) \
                    .reduceByKey(lambda a, b: a + b)
        wordcount.take(10)

Out[3]: [('The', 4460),
         ('Project', 79),
         ('of', 16830),
         ('Shakespeare', 5),
         ('', 263644),
         ('is', 8334),
         ('use', 289),
         ('anyone', 8),
         ('anywhere', 5),
         ('in', 10759)]

In [ ]:
```

Figure 1: Jupyter notebook with Spark cell setup and Word Count execution



Spark Jobs (?)

User: Linh B Ngo
Total Uptime: 30 min
Scheduling Mode: FIFO
Completed Jobs: 2

Completed Jobs (2)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	runJob at PythonRDD.scala:154 runJob at PythonRDD.scala:154	2022/09/08 20:54:10	3 s	2/2	3/3
0	runJob at PythonRDD.scala:154 runJob at PythonRDD.scala:154	2022/09/08 20:42:35	3 s	2/2	3/3

Page: 1 — 1 Pages. Jump to 1 . Show

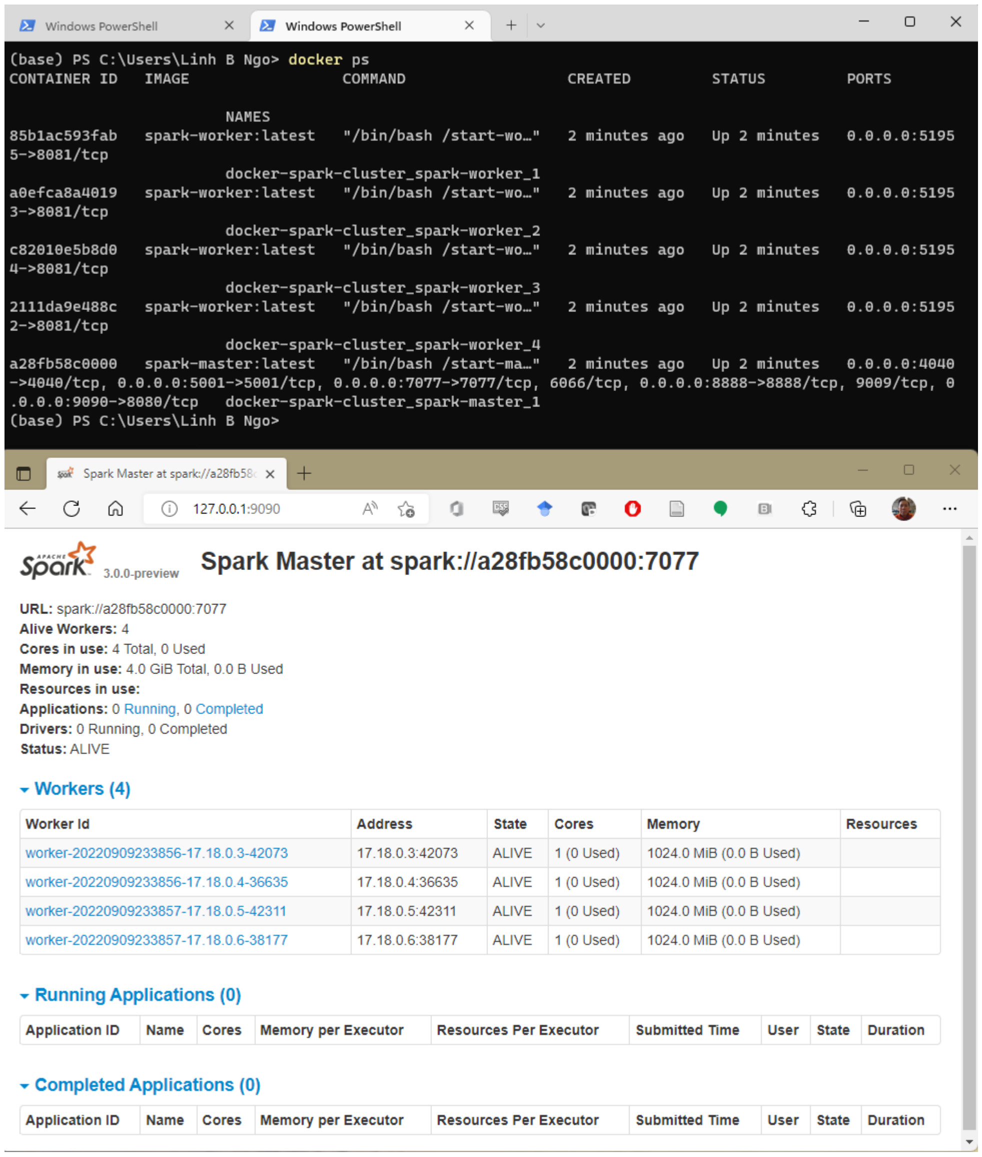
Figure 2: Spark Web UI on local server

and PySpark [13] is necessary to avoid overhead that could happen from containerized solutions. The trade-off in this case is the added complexity of managing various installations in a Windows environment. For more powerful devices, single-instance Docker solutions can be used, where all required libraries for Spark and Python are already configured inside the container. With top-of-the-line PCs, we can also deploy a multi-node Spark cluster that is housed in multiple containers. In all scenarios, we are assuming Windows installation as it is the most popular operating system used by students.

For direct deployment, the following components must be setup: 1) Anaconda [5], 2) Apache Spark, 3) Java, and 4) Hadoop-Windows utilities. Out of these four components, Anaconda potentially takes up the most space (approximately at least 500MB) and requires an installation process. While it is possible to selective pick only relevant Python components necessary to support Apache Spark, the steps will be lengthy and tedious. Students lacking command-line experience and administration skill will likely encounter errors, creating technical overhead inside and outside of the classroom. Java can either be installed or decompressed to a specific location. Apache Spark and Hadoop-Windows utilities need to be downloaded and decompressed to specific locations. Once everything is in place, environment variables need to be set for *ANACONDA_HOME*, *SPARK_HOME*, *HADOOP_HOME*, and their corresponding sub-directories to executable files in *PATH* via Windows' System Properties.

After a Jupyter notebook is created, a block of template code is provided to setup the launching of a local Spark cluster. This template includes getting the location of Spark's installation via *SPARK_HOME* and append relevant supporting libraries to the notebook's Python kernel. Students can specify the size of the cluster via number of cores and amount of memory in GB. Finally, *PySpark* will launch the local Spark cluster. Figure 1 demonstrates the execution of the template cell, and the subsequent running of another cell that runs the word count activities and returns the top ten unique words' counts. Figure 2 shows the records of the submitted Spark jobs on the local 127.0.0.1:4040 address.

In both single-node [2] and multi-node [1] containerized deployments, the key setup step is to install Docker Desktop. The challenge is the enabling of virtualization support on older laptop models via BIOS. This issue has gradually been reduced over time as newer laptops have virtualization enabled by default. The deployments launch the Docker container(s) and expose the default port of the Jupyter notebook server to the host machine, making Jupyter available to students via the host browser. One downside of this approach is the limited access to Spark's Web UI. While it is possible to expose the primary interface of the Web UI, additional log information resides on individual Spark worker's container whose port must be exposed separately. It is possible to examine the log from the terminal using *docker log* command. However, this creates a potential point of failure/technical overhead for students. An example multi-node containerized deployment is shown in Figure 3.



```
(base) PS C:\Users\Linh B Ngo> docker ps
CONTAINER ID   IMAGE                 COMMAND                    CREATED          STATUS         PORTS

                      NAMES
85b1ac593fab   spark-worker:latest   "/bin/bash /start-wo…"   2 minutes ago    Up 2 minutes   0.0.0.0:5195
5->8081/tcp
                      docker-spark-cluster_spark-worker_1
a0efca8a4019   spark-worker:latest   "/bin/bash /start-wo…"   2 minutes ago    Up 2 minutes   0.0.0.0:5195
3->8081/tcp
                      docker-spark-cluster_spark-worker_2
c82010e5b8d0   spark-worker:latest   "/bin/bash /start-wo…"   2 minutes ago    Up 2 minutes   0.0.0.0:5195
4->8081/tcp
                      docker-spark-cluster_spark-worker_3
2111da9e488c   spark-worker:latest   "/bin/bash /start-wo…"   2 minutes ago    Up 2 minutes   0.0.0.0:5195
2->8081/tcp
                      docker-spark-cluster_spark-worker_4
a28fb58c0000   spark-master:latest   "/bin/bash /start-ma…"   2 minutes ago    Up 2 minutes   0.0.0.0:4040
->4040/tcp, 0.0.0.0:5001->5001/tcp, 0.0.0.0:7077->7077/tcp, 6066/tcp, 0.0.0.0:8888->8888/tcp, 9009/tcp, 0
.0.0.0:9090->8080/tcp   docker-spark-cluster_spark-master_1
(base) PS C:\Users\Linh B Ngo>
```

3.0.0-preview **Spark Master at spark://a28fb58c0000:7077**

URL: spark://a28fb58c0000:7077
Alive Workers: 4
Cores in use: 4 Total, 0 Used
Memory in use: 4.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (4)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220909233856-17.18.0.3-42073	17.18.0.3:42073	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20220909233856-17.18.0.4-36635	17.18.0.4:36635	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20220909233857-17.18.0.5-42311	17.18.0.5:42311	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20220909233857-17.18.0.6-38177	17.18.0.6:38177	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Figure 3: Multi-node containerized deployment of Spark cluster

2.2 Scaling to community cloud

The containerized solutions for single and multi-node Spark cluster can also be used to deploy at scale on CloudLab, a federal cloud resource that is available for research and education purposes [21]. A CloudLab experiment needs to be deployed prior to class. This experiment launches a single Docker node or a multi-node Docker Swarm [3]. Students can launch the containerized deployments here on the experiment and access the Jupyter server via the public IP address of the experiment's head node.

As CloudLab is designed to be an experimental test-bed, cloud allocations are provisioned within limited timing duration (16 hours) that are not suitable for sustained learning activities. However, long-term availability of personal computing devices can be combined with CloudLab to create a learning model that enables students to build their big data workflow locally using smaller data set and test their solutions on CloudLab using larger data sets.

The combination of personal computing devices and public cloud resources facilitates sustainable and scalable solutions to provide learning environments for BDC topics. In the next section, we will discuss learning activities that are created to support this approach.

3 LEARNING ACTIVITIES AND ASSESSMENT

While designing learning activities, we focus on guiding students through fundamental steps in the big data processing pipeline. These steps include identifying data sources, acquiring and ingesting raw data, and analyzing curated data. Students are exposed to both the underlying theory behind big data techniques and the software tools and infrastructures that implement and support these techniques. Assessments include both short-term assignment and a semester-long data analysis project.

3.1 Learning activities

Data acquisition: The first topic of interest is data identification, acquisition, and curation. The experience of combing through a large number of data sets can be rewarding but also can be frustrating at times. For this topic, students learn to narrow down their topic area and not to focus solely on quantity but on quality of the data sets being examined. Qualities such as cleanliness, reliability and uniqueness have a direct impact on subsequent learning activities. There are many publicly available data sets spanning across many topic areas for students to explore. Examples include *data.gov* [10] for data related to government, climate, health, energy, and economy, *Kaggle* [16] for health, science, sports, crypto, and entertainment data, *UCIML* [26] for science and engineering data, *Nasdaq Datalink* [20] for financial and business related data. Table 1 shows a list of sources and topic areas for publicly available data sets.

Additionally, we also make security data collected from our own Linux servers available to students. For larger data sets, smaller samples that can easily be processed on PCs are generated.

Programming models: Existing big data toolkits (e.g., Hadoop [12], Spark [28], AllPairs [19], DataSpaces[25], etc) already provide an extensive collection of ready-to-use functionalities. It is critical that students understand the underlying programming paradigms implemented in these functionalities. They are to momentarily step away from the traditional procedural and object-oriented paradigms where functions and objects are the targets of programming activities. Instead, they are to focus on the data pipelines and how these pipelines will eventually produce the desired results. The MapReduce programming paradigm [11] is one such dataflow programming paradigm, where programmers utilize ‘map‘ and ‘reduce‘ functions to form the data pipelines. This paradigm is implemented in Hadoop MapReduce, Apache Spark, and many other big data frameworks.

3.2 Assessments

In addition to the standard quizzes and exams that assess students' on their understanding of foundational concepts, assignments and semester projects are key components to the assessment process. The following assignments and projects were disseminated to students and carried out primarily using the infrastructures described in Section 2.

Assignments: Assignments are used for WCUPA's BDC courses. There is a total of five assignments that work on progressively bigger and more complex data sets. Students demonstrate their understanding by implementing well-known algorithms such as PageRank and K-mean clustering using MapReduce programming paradigms and apply them on the data.

- Assignment 1 is a straightforward demonstration that students are able to deploy Spark on their PC/laptops. This assignment requires students to provide a series of screenshots showing working Jupyter notebooks, Spark WebUI, and success WordCount results. The assignment serves as a confirmation that all students have access to adequate infrastructure to continue the course.
- Assignment 2 provides students with an actual security log of a public-facing computer. Students are to study the log and provide answers to the following questions: 1) How many failed access attempts? 2) Which countries these attempts are generated from? 3) What are the attempted usernames? 4) Which date has the highest attack frequencies? These questions require students to become familiar with Spark's actions and transformations and also to learn how to examine complex textual data.
- In Assignment 3, students are first introduced to a *big* data set: the user information portion of Yelp's academic data set [4]. This data set is approximately 1.8Gb compressed, which is large enough to be inconvenient. Besides descriptive statistics, students are required to identify the top ten influential users from this data set. This particular requirement is open-ended, as students will need to justify their choice of attributes that define level of influence.
- Assignment 4 is an extension of assignment 3, where students now use all data within the Yelp data set (user, review, and business) to study characteristics of the influential users and identify their pattern of restaurant visits (local, regional, or east-west coasts). This assignment is where students can decide to apply complex techniques such as PageRank and K-mean clustering.
- Assignment 5 introduces students to Kaggle [6]. The assignment involves two parts. In part 1, students are to participate in the introductory "Titanic - Machine Learning from Disaster" competition but use Spark and its supporting libraries to carry out the prediction task. In part 2, students study the cryptocurrency data set on Kaggle. While this data set is not overly large (approximately 300Mb), the text line themselves contains non-standard characters and are not easily tokenized.

Semester Project: We have students design and implement a data analysis workflow to analyze the data set of their own choice. The goal of this activity is to get students ready to apply what they learn to work on real-world problem beyond the classroom. We ask students to come up with at least five interesting questions they want to answer from the chosen data set. If the student decides MapReduce programming model is suitable to answer those questions, the student would write their own mapper and reducer functions specific to each question. A prototype of the workflow is developed on small sampled data and run on PCs. Later, this prototype is migrated to run on a large-scale BDC environment that could be on-site or cloud-based, depending on the institution' resources. We evaluate the entire workflow by processing different data sizes and scaling out across different number of computers and whether the results support answering the proposed questions.

Case Study: One of our students wanted to perform a study of educational data, specifically,identifying the correlation between a student's gender, course work performance and the likeliness of he/she pursuing a career in STEM after high school graduation. The student looked into a number of educational data sets from the National Center for Education Statistics (NCSE) and chose a

Table 1: Popular sources for data sets

Source	Topic Areas
data.gov	Government, Climate, Health, Energy, Economy,...
Kaggle	Health, Science, Sports, Crypto, Entertainment,...
UCI ML Repository	Life Science, Physical Science, Engineering,...
Nasdaq Datalink	Financial, Real Estate, Banking,...

data set from the High School Longitudinal Study[27]. The study surveyed over 20,000 students from more than 900 both public and private schools.

After obtaining a data set, the next step was for the student to propose a list of questions they want to investigate. The questions are listed in Table 3. As an example of the type of insight student could derive from the data set, he/she propose a hypothesis which state that a student with higher GPA is more likely to take post-secondary classes (college/university) because GPA is a good indication of the student's future academic aspiration in higher education. The student then proceed with creating customized mapper and reducer functions to attempt to validate the hypothesis. Figure 4 shows a part of a mapper code.

```
if(gpaNum == -10) {
    gpaRange = "error␣parsing";
}
else if(gpaNum < 0 && gpaNum != -10){
        gpaRange = "missing␣GPA";
} else if (gpaNum >= 0 && gpaNum < 0.5 ){
        gpaRange = "0.0-0.5";
}else if (gpaNum >= 0.5 && gpaNum < 1.0 ){
        gpaRange = "0.5-1.0";
} else if(gpaNum >= 1.0 && gpaNum < 1.5){
        gpaRange = "1.0-1.5";
} else if(gpaNum >= 1.5 && gpaNum < 2.0){
        gpaRange = "1.5-2.0";
}else if(gpaNum >= 2.0 && gpaNum <2.5){
    gpaRange = "2.0-2.5";
...
}
```

Figure 4: A snippet of a customize mapper function

The analysis result validated the hypothesis. Table 2 shows as the GPA increases, there are more student answered yes to indicate they were talking postsecondary courses after graduating from high school.

4 DISCUSSION

The deployment varieties on PC presented in Section 2 have been used in one BDC course from West Chester University of Pennsylvania (WCUPA) and a series of independent study course from Western Illinois University (WIU). Both institutions are regional public universities and lack either infrastructure (WCUPA) or personnel support (WIU) to deploy and maintain large-scale computing infrastructures for regular BDC courses. The adoption of the above-mentioned deployment varieties enable teaching and learning activities of BDC topics and subsequent course creation.

Table 2: Student response to the questions about taking post-secondary courses

GPA	Yes	No	Don't know
0.0-0.5	32	58	49
0.5-1.0	70	158	110
1.0-1.5	195	343	189
1.5-2.0	590	618	319
2.0-2.5	1299	708	329
2.5-3.0	2073	617	269
3.0-3.5	2449	417	169
3.5-4.0	2768	202	70
4.0+	3390	99	16

4.1 Course descriptions

At WCUPA, there was no BDC course prior to 2019. A special topic course on complex large-scale systems was offered with emphasis on BDC contents during the Winter 2019 semester. During this initial offering, the multi-node containerized local deployments were introduced to the course. At this time, Docker required Docker Toolkit and VirtualBox to support containerization for older versions of Windows and Mac, and several students with much older machines could not run a multi-node solution. A single-node solution was developed and introduced as an alternative. After the initial offering in Winter 2019, the course was offered once again as a special topic course in Winter 2020. During this semester, the direct deployment approach was introduced to students. Instructions for Docker-based deployments were made available to students as alternatives but were not formally introduced in class. Beginning Fall 2021, the course was converted into a regular Fall-semester course titled “Big Data Engineering."

Because traditional BDC courses are not offered at WIU, students who want to gain experience working with big data often do so through an independent study course. The course includes most of the activities outlined in Section 3 culminating with a semester long final project. This specific independent study course has been taken by WIU graduate students four times in recent years.

4.2 Student outcomes and feedback

Descriptive summary of student enrollments during for BDC course offering at WCUPA is presented in Table 4. It should be noted that for the Fall 2022 semester, class size was capped at 35 due to limited physical seating. In general, students' verbal feedback has been positive. For Winter semesters, there were no formal evaluation

Table 3: Proposed questions by a student [18]

#	Questions
1	Does a higher high school GPA correlate to enrollment in a post-secondary education program towards earning a bachelor's?
2	Does a higher high school GPA correlate to the number of STEM courses enrolled in during high school?
3	Is there a correlation between gender and the decision to enroll in a post-secondary educational program?
4	Is there a correlation between gender and the amount of STEM courses taken during high school?
5	Does a positive response about the value of math also correlate with the number of science related courses enrolled in during high school?
6	How do different genders perceive the importance of math and their own ability to do well in math compared to that of the other gender?

Table 4: BDC Course offerings and enrollments at WCUPA.

Semester	Enrollment
Winter 2019	13
Winter 2020	20
Fall 2021	40
Fall 2022	35

process. However, in-class interactions had been positive and no significant technical issues happened. During Winter 2020, two students decided to follow up on the security analysis assignment and expanded the work to cover all system logs of department computers. This resulted in a publication that won Best Student Paper award at a regional conference [9]. For the first regular offering, students' ratings of course contents have been well-above departmental, college, and university's mean rating. As Fall 2021 was the first semester back after Covid-19, no verbal feedback from students was made available.

4.3 Lesson learned

We offer the following lessons to summarize our experience in teaching big data computing to both undergraduate and graduate students. We hope these can be applied to sustain and improve teaching and learning experience for fellow educators and students in the future.

Keep sustainability and reproducibility at the forefront: Limited access to workable resources is the primary obstacle to teaching and learning BDC, or PDC for that matter. Helping students to deploy a suitable computing environment on their PCs ensures access and allows students to at least learn about the underlying theories, even without scalability demonstration.

Give students some autonomy: Having access to the computing environment on their own PCs will also contribute to students' autonomy in learning and experimenting with various data sets from topics that they are interested in. Many data sources available on 'data.gov' are actually small in size (dozens of megabytes) and well suited for PC processing. Working with data sets in topic areas that they can relate to either their future career or their personal hobbies can help boost students' motivation.

Be ready to clean data: Ideally, students are able to find a useful and clean data set that contains all relevant information. However, this is almost always not the case, and students should be reminded to be mindful about the cleanliness and trustworthiness of their collected data. Data cleaning is an important part of the data analysis workflow. Additional data sets might be needed before the final data product is ready to be analyzed.

Start small, then scale-up: Students can create a small sample from a big data set while maintaining original statistical properties. They would then design, implement, and deploy an end-to-end analysis prototype workflow for the small sample on their PC environment. A scale-up deployment on the large-scale resource is carried out later. One of the side advantage of starting small is failing faster. If something go wrong, students would have chances to alter their design and implementation and rerun their experiments. Once they are confident on the correctness of their analysis, they can deploy to the large-scale resource for final validation and performance evaluation.

Use high level abstractions to speed up progress: Although we spend time on low level programming paradigms to provide necessary foundational knowledge on BDC, modern frameworks such as Apache Hadoop or Apache Spark provide students with more suitable functionalities to build their data analysis workflow. It is important for students' learning growth that they can step away from the low-level paradigms and focus on designing an appropriate dataflow pipeline. The question then becomes, what functionalities would they need to *shape* their pipeline. In doing so, students can leave complex decisions such as coordination and synchronization among multiple compute nodes to the framework.

5 CONCLUSION AND FUTURE WORK

In this paper, we present our approach to create a sustainable and scalable approaches in setup personal computing environment for big data computing. Our approach potentially can free educators from tedious tasks of maintaining distributed computing infrastructure. Instead, they can focus on teaching and mentoring activities. For students, they will have access to resources that can be recreated and duplicated outside classrooms, enabling self-learning beyond the scope and duration of the class. We also discuss a list of active learning activities that play a large role in achieving many important learning objectives.

With the gradual changes in Windows' toward supporting Linux environment and better yet inexpensive computers, additional work needs to be done to continue improving the above approaches. These include, but not limited to

- Creating better documentation for the direct deployment process. Video instructions might be more useful than static documentation.
- Convert the multi-node Docker deployment from using Docker Compose to using Kubernetes [8]. This allows this approach

to be deployed as a centralized infrastructure if a cloud resource becomes available.

- Improve the Docker deployment to make external data and code integration more dynamic. This will reduce complexity of importing/uploading external materials into the containerized environment.

REFERENCES

[1] 2022. Docker compose file for Multi-node Spark Server. https://github.com/linhbngo/docker-spark-cluster

[2] 2022. Docker image for Single-node Spark Server. https://github.com/linhbngo/docker-images/tree/master/csc467

[3] 2022. Docker profile to launch Docker and Docker Swarm. https://github.com/CSC468-WCU/csc468cloud/tree/docker

[4] 2022. Yelp Open Dataset. https://www.yelp.com/dataset

[5] Anaconda. [n. d.]. Anaconda: Open-source Python distribution platform.

[6] Casper Solheim Bojer and Jens Peder Meldgaard. 2021. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting* 37, 2 (2021), 587–603.

[7] Richard A Brown. 2009. Hadoop at home: large-scale computing at a small college. In *Proceedings of the 40th ACM technical symposium on Computer science education*. 106–110.

[8] Brendan Burns, Joe Beda, Kelsey Hightower, and Lachlan Evenson. 2022. *Kubernetes: up and running*. " O'Reilly Media, Inc.".

[9] Tyler Clark, Kevin Codd, and Linh B. Ngo. 2021. Studying break-in attempts across multiple servers using Apache Spark and security logs. In *IFIP International Conference on Network and Parallel Computing*. Annual Spring Conference of the Pennsylvania Computer and Information Science Educators.

[10] data.gov. 2022. The home of the U.S. Government's open data. https://www.data.gov

[11] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113.

[12] Jens Dittrich and Jorge-Arnulfo Quiané-Ruiz. 2012. Efficient big data processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment* 5, 12 (2012), 2014–2015.

[13] Tomasz Drabas and Denny Lee. 2017. *Learning PySpark*. Packt Publishing Ltd.

[14] Joshua Eckroth. 2016. Teaching big data with a virtual cluster. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 175–180.

[15] Jesse Eickholt and Sharad Shrestha. 2017. Teaching big data and cloud computing with a physical cluster. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 177–181.

[16] Kaggle. 2022. Kaggle Open Datasets. https://www.kaggle.com/datasets

[17] Henry Kasim, Verdi March, Rita Zhang, and Simon See. 2008. Survey on parallel programming model. In *IFIP International Conference on Network and Parallel Computing*. Springer, 266–275.

[18] Corbett Megan. 2016. Utilizing the Apache Hadoop MapReduce to Analyze Trends in High School Students' Performance and Postsecondary Decisions. *Independent Study Report* (2016).

[19] Christopher Moretti, Hoang Bui, Karen Hollingsworth, Brandon Rich, Patrick Flynn, and Douglas Thain. 2009. All-pairs: An abstraction for data-intensive computing on campus grids. *IEEE Transactions on Parallel and Distributed Systems* 21, 1 (2009), 33–46.

[20] Nasdaq. 2022. Nasdaq Datalink. https://data.nasdaq.com/

[21] Linh B Ngo and Jeff Denton. 2019. Using CloudLab as a Scalable Platform for Teaching Cluster Computing Ambassador Program. *The Journal of Computational Science Education* 10, 1 (2019).

[22] OpenMP. 2022. The OpenMP API specification for parallel programming. https://www.openmp.org.

[23] OpenMPI. 2022. A High Performance Message Passing Library. https://www.open-mpi.org

[24] Ariel S Rabkin, Charles Reiss, Randy Katz, and David Patterson. 2012. Experiences teaching MapReduce in the cloud. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 601–606.

[25] Melissa Romanus, Fan Zhang, Tong Jin, Qian Sun, Hoang Bui, Manish Parashar, Jong Choi, Saloman Janhunen, Robert Hager, Scott Klasky, et al. 2016. Persistent data staging services for data intensive in-situ scientific workflows. In *Proceedings of the ACM International Workshop on Data-Intensive Distributed Computing*. 37–44.

[26] UCI. 2022. UCI ML Repository. https://archive.ics.uci.edu/ml/index.php

[27] USDE-NCES. 2016. The School Longitudinal Study, 2009-2013 United States.

[28] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. 2016. Apache spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65.