Sanzu: A Data Science Benchmark

Alex Watson, Deepigha Shree Vittal Babu, and Suprio Ray
Faculty of Computer Science, University of New Brunswick, Fredericton, Canada.
Email: awatson@unb.ca, dvittal@unb.ca, sray@unb.ca

***Abstract*—The volume of data that is generated each day is rising rapidly. There is a need to analyze this data efficiently and produce results quickly. Data science offers a formal methodology for processing and analyzing data. It involves a work-flow with multiple stages, such as, data collection, data wrangling, statistical analysis and machine learning. In this paper, we look at data analytics systems that support the data science work-flow. The variety of current commercial and open-source data analytics systems differ significantly in terms of available features, functionality, and scalability. A benchmark can be used to evaluate the functionality and performance of a system. However, there is no standard benchmark for evaluating or comparing these data systems for doing data science. In this paper, we introduce a data science benchmark, Sanzu, to evaluate systems with data processing and analytics tasks. Our benchmark includes a micro and macro benchmark. The micro benchmark tests basic operations in isolation. It consists of task suites for reading and writing, data wrangling, statistical analysis, machine learning and time series analysis. Each macro workload evaluates an analytics application where a series of analysis or functions are based on a real world application. The macro benchmark focuses on sports and smart grid analytics. We evaluate these tasks on five different popular data science frameworks and systems: R, Anaconda Python, Dask, PostgreSQL (MADlib) and PySpark. For micro benchmark we generate synthetic datasets with 3 scale factors: 1, 10 and 100 (scale factor 1=1 million). The macro benchmark uses data generated from real-world data sources.**

I. Introduction

The digital information revolution is accelerating the growth of data. Data is everywhere and a significant part of our daily activities generate data. These activities include online shopping, browsing and search, and even offline tasks like medical checkup at a clinic or check-in at our local grocery stores. Besides human activities, a vast amount of data is generated from machine to machine (M2M) interaction. However, most of the collected data is raw. To extract value from the data, it is necessary to process and analyze the data to enable the creation of actionable insights. With advances in data collection and storage technologies, data analysis has become widely accepted as the fourth paradigm of science [12].

Data science provides formal methods and techniques for processing and analyzing data. Data science is evolving as a field and it derives skills from different disciplines, such as math, statistics and computer science. The data science process is characterized by a workflow with a feedback loop (described in Section III-B). In this workflow different tasks are pipelined, which resemble the stages in the "big data pipeline" as described in [14]. However, data science not only includes notions of big data technical challenges, but also considerations that might arise even with smaller datasets [3].

To derive value from raw data, it needs to be cleaned, analyzed, manipulated, stored and retrieved. These actions can be performed in different ways. Historically, data was stored in the database, retrieved when needed and then operations were performed on it. This can be very useful in many situations. However, when there are large amounts of data that need to be analyzed together and multiple times, this can result in inefficiencies. Some of these inefficiencies are due to constantly retrieving and storing data from the database. The others are due to a lack of support for one or multiple stages of the data science workflow. To address these challenges, a number of data systems and frameworks have been developed to analyze data. These data systems, libraries and platforms, are constantly evolving. The variety of systems offered by current commercial and open-source data analytics systems, differ significantly in terms of available features, functionality and storage capacity.

In computing, the purpose of a benchmark is to run one or more computer programs to assess the relative performance of an object, by running a number of standard tests against it [4]. Benchmarks have been developed to evaluate some of the big data systems ([1], [6], [5], [8], [29], [15]). They focus on either business model queries or more specific tasks and implementations, such as sorting, genomics algorithms and smart grid analytics. However, to our knowledge, there is no benchmark to evaluate the functionality and performance of the systems for doing data science. It is unclear what system does this best because there is no industry standard for evaluating or comparing the data science systems that are commonly used for these kinds of analysis.

To address this need we introduce a data science benchmark called Sanzu. Our benchmark is intended to serve as a basis for an industry standard benchmark for evaluating systems and libraries for doing data science. The benchmark includes two components: a micro benchmark and a macro benchmark. The micro benchmark consists of several task suites that closely follow the stages in the data science workflow, including data loading, data wrangling, descriptive and inferential statistical analysis and machine learning. For instance, Sanzu looks at the capacity for the systems to do simple statistical tasks, such as filtering and finding the central tendency, to more complex tasks with machine learning, such as linear regression and K-means Clustering. It also includes time series tasks owing to its importance in different industries. The intent of the macro benchmark part of Sanzu is to incorporate applications that are representative of real-world data science use cases. To that end, the macro benchmark includes some real-world tasks in the field of sport analytics, particularly, ice hockey. It also includes a smart grid analytics application, with a focus on demand curves. Sanzu is an open-source project and the source code is publicly available [1].

We use Sanzu to evaluate five popular data science frameworks and systems. The five data analytic systems are R [24], Anaconda Python [23], Dask [2], PostgreSQL (MADlib) [22] and PySpark [28]. The research focus of this paper is on the performance of each individual system completing each particular task. The benchmark aims to expose strengths of the systems as well as particular limitations of each system. The micro benchmark tasks were first evaluated with a small dataset to ensure consistency of results across all systems. Then they were tested with three different datasets by varying the scale factor (SF: 1, 10, and 100, where SF 1=1 million rows) to evaluate their scalability.

While developing this benchmark we have found that some of the systems lack support for specific tasks. So wherever we have specified "NF" (No Functionality) in the paper, we do not claim that it is impossible to implement it in a given system. But rather, we have found no ready to use built-in functionality. We are also aware that while choosing some built-in functionality we may have excluded some optimization. Finally, we have discovered that several systems do not scale with larger datasets (scale factor 10 or 100). We believe that our findings will open up ongoing discussion within the community, which may lead to improved support for functionality and scalability for the data science tasks within some of the systems. It is our hope that our effort would help in the adoption of a benchmark that will be widely used by the data scientists.

II. Related Work

To our knowledge, no benchmark exists that can be termed as a data science benchmark. However, there are several categories of benchmarks that are related. Relational database benchmarks belong in the first category. In the second category are the big data systems benchmarks. Then, there are domain specific analytics benchmarks; but these tend to focus on specific tasks, such as smart grid or genomics problems.

A. Database benchmarks

There are a number of commercially available and free database benchmarks. The most well known is the Transaction Processing Performance Council (TPC) [30]. It is devoted to defining transaction processing and database benchmarks for diverse workloads. TPC-C is an On-line Transaction Processing (OLTP) benchmark to measure transactional throughput. TPC-H is a decision support benchmark characterized by the execution of complex SQL queries. Some of these benchmarks have evolved over the years. TPC-DS is TPC's latest decision support benchmark with a complex snowflake-like data model.

Besides transactional and decision support workloads, there have been attempts to develop database benchmarks for other kinds of workloads. For instance, Jackpine [25] is a benchmark to evaluate spatial database performance.

[1]Sanzu data science benchmark: http://bigdata.cs.unb.ca/projects/sanzu

B. Big data systems benchmarks

The growing popularity of big data systems, such as Hadoop [10] and Spark [28], led to the development of several benchmarks. These benchmarks address the different aspects of big data: volume, variety and velocity. YCSB [1] is arguably the first among the big data benchmarks. The focus of this benchmark was to evaluate the performance of emerging NoSQL data stores and also to compare them against traditional relational databases. In the original paper, the authors executed three different workloads with four different data systems: HBase, Cassandra, PNUTs, and MySQL. TeraSort (or GraySort) [8] is another well-known benchmark to be run on commercially available hardware. It is a micro benchmark that sorts a large number of 100-byte records with the goal of stress testing processing, and storage I/O subsystems. BigBench [6] is a popular end-to-end big data benchmark. It provides a data model and synthetic data generator. Its data model was adopted from the TPC-DS benchmark, and extended that with semi-structured and unstructured data components. Recently, a modified version of BigBench, called BigBench V2 [5], has been introduced. It includes a new data model and an overhauled workload, that includes Web-logs modeled as key-value pairs.

C. Domain specific analytics benchmarks

Mehta et al. [17] evaluated several systems with scientific image data processing tasks. The uses cases were from neuroscience and astronomy. The systems they evaluated were Spark, Myria, Dask, SciDB and TensorFlow. In [29] a genomics benchmark, GenBase, was developed to measure systems performance on data managaement and data analytics tasks. The systems that were evaluated were R, PostgreSQL, a column store database, SciDB, and Hadoop. A benchmark to evaluate common smart grid analytic tasks was developed by Liu et al. [15].

III. Motivation

In this section, we discuss the motivation behind our work.

A. Data science

Data science provides tools, techniques and methodologies to turn data into insight. It is still evolving as a field. Joel Grus has defined a data scientist as someone who extracts insight from messy data [9].

B. Data science workflow

Data science draws various techniques and skills from different fields, including math and statistics, machine learning and computer science. In addition, a data scientist may possess domain expertise in her own areas. Invariably, data science tasks follow a series of steps, from data gathering to building analytical models. The data science process [26], can be conceptualized as a workflow.

As shown in Figure 1, the first step of the workflow is the collection of data. The second step is data wrangling, which involves loading, cleaning, transforming, and rearranging messy

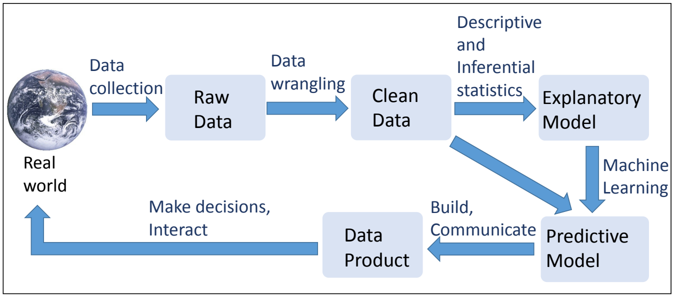


Fig. 1. Data Science Workflow

Software	Version	Language	Key libraries
Anaconda	4.4.0	Python	Pandas, Numpy, MatplotLib, Scikit
Dask	0.15.1	Python	
PySpark	2.11	Python	Hadoop 2.3
PostgreSQL	9.6	SQL	MADlib
R	3.4.1	R	

TABLE I
SOFTWARE CONFIGURATION

data for easy access and analysis. This data preparation step is vital for subsequent downstream processing. According to some estimates [20], on average about 50% to 80% of the time in the data science workflow is spent in data wrangling. The next step is to apply statistical analysis on the clean data to build explanatory model. The statistical analysis could be descriptive or inferential. Descriptive statistics quantitatively describes the characteristics of a dataset. Essentially, it is used to summarize the entire population. Important descriptive statistical properties include measures of central tendency and measures of dispersion. Inferential statistics is used to make generalization about the population based on representative subsets of the population called samples. A common approach to investigate a claim about the population by analyzing samples is hypothesis testing. The statistical analysis techniques can be used to build explanatory model. The explanatory model establishes a causal effect from the observed data. On the other hand, the predictive model is used to predict new observations and can be constructed by machine learning algorithms. The choice of a particular machine learning algorithm depends on the type of problem at hand.

Once there is a model we could communicate our results or build a "data product", such as an Email spam filter. The data product is usually incorporated back into the real world, thus enabling user interaction with that product. This process helps generate more data, and in turn creates a feedback loop. Schutt and O'Neil [26] noted that this feedback loop is a key distinguishing feature of the data science process.

C. *Why a data science benchmark?*

We believe that a benchmark for data science should faithfully follow the data science process. Reflecting the ground reality, it should contain tasks from every step of the workflow (as in Figure 1). As described in Section II, there are a few benchmarks that focus on a few domain specific analytics applications. However, no benchmark exists that captures the data science process. Given the growing importance of data science, we have developed our benchmark. We call it "Sanzu", which is a mythical river of life in Japanese tradition, to symbolize the workflow in data science.

IV. SYSTEMS EVALUATED

In this section, the five analytics systems that we have evaluated are described. They are all publicly available open-source software. The software version, programming language and key libraries of these systems are shown in Table I. We chose these five analytic systems because we believe that they are a good representation of current state-of-the-art tools used in data science. Moreover, they were used in other benchmarks mentioned in Section II. PostgreSQL is a popular relational database with support for machine learning in SQL (MADlib) and it has been used by other benchmarks such as [25], [29] and [15]. R is one of the most popular statistical computing libraries and is also used in other benchmarks ([29]). Anaconda is the most popular Python-based data science platform. Dask extends Anaconda and adds support for out-of-core execution and parallelism to some of the libraries within Anaconda. It was used in the benchmark in [17]. Spark is one of the leading candidates among the big data systems with its distributed data processing framework and it was used in a few benchmarks ([17], [5] and [15]). Next we briefly describe each of the five systems.

1) Anaconda: Anaconda is an open data science platform, with a collection of over 720 open source packages [23]. The source packages that are utilized in this benchmark are: Pandas, NumPy, Matplotlib and scikit-learn. Pandas is a package that provides data structures and tools for data analysis [21]. NumPy is a package for scientific computing with Python. With a powerful N-dimensional array object, it supports linear algebra, and other capabilities [19]. Matplotlib is a Python plotting library to produce figures and graphs [16]. Scikit-learn is a machine learning library that is built on NumPy [27].
2) Dask: Dask [2] is a parallel computing library for analytic computing. It is composed of two components: dynamic task scheduling and big data collections. This collection includes parallel arrays, dataframes, and lists that extend common libraries like NumPy and Pandas. Dask represents computation as task graphs with data dependencies and it supports out-of-core execution.
3) PostgreSQL (with Apache MADlib): PostgreSQL [22] is an open-source object relational database management system. PostgreSQL is accompanied by MADlib [11], an in-database machine learning toolkit which performs statistics and analytics. It provides many SQL-based algorithms for machine learning, data mining and statistics which run at scale within a database engine.
4) PySpark (with Hadoop): PySpark is the Python API from Apache Spark [28]. Spark is a main-memory distributed

data processing platform. It provides an application programming interface centered on a data structure called the resilient distributed dataset (RDD). An RDD is a read-only multiset of data items that can be distributed over a cluster of machines. Spark supports data manipulation, statistical analysis and machine learning analysis.

5) R: R is a language and environment for statistical computing and graphics [24]. R has the ability for data handling and storage facility, data manipulation, statistical analysis, time series analysis, and machine learning algorithms. R is extensible and includes a wide collection of open-source packages.

V. The Benchmark

Our benchmark, Sanzu, consists of two different components: a micro benchmark and a macro benchmark. The micro benchmark consists of several task suites, which involve data wrangling, statistical, time series and machine learning analyses. The macro benchmark contains two applications that are modelled after real-world use cases.

A. Micro benchmark

The micro benchmark is intended to test the basic tasks across multiple data analytics systems. These tasks are common and are used everyday in solving real-world and industry problems. They are modeled after the data science workflow (in Section III-B). The micro benchmark consists of 6 task suites: basic file I/O, data wrangling, descriptive statistics, distribution and inferential statistics, time series analysis and machine learning. Each suite represents a different stage in the data science workflow. Each suite includes a number of tasks. Usually, some of the operations involving statistical distribution, such as correlation and skew, are considered as part of descriptive statistics. But we put them under the task suite distribution and inferential statistics, to have a sizable number of tasks in that suite. Ideally, these tasks should be based on built-in functions or should have simple implementation in each of the systems. We implemented the tasks using built in functionality when available. However, in some cases this was not possible. The tasks included in the micro benchmark are summarized in Table II. The table also indicates the level of support for these tasks in each data system.

1) Micro benchmark task suites:

- **Basic File I/O**: An important goal of the micro benchmark is to demonstrate weaknesses and/or strengths on a specific task for each of the systems. File I/O operations are the basic tasks that are necessary to do any data analysis. In our benchmark reading and writing operations are performed with CSV files. This file format was chosen for consistency, because each data system has the ability to read and write from CSV.
- **Data Wrangling**: The data wrangling tasks are: sorting, filtering, merging, group by and removing duplicates. All the five data systems provide built-in functionality to implement all of these tasks.
- **Descriptive Statistics**: The descriptive statistical analysis tasks are: finding central tendency, finding measures of dispersion, ranking, eliminating outliers and plotting a scatter plot. However, the outcome of breaking ties in sorting is different depending on the default data system. Dask does not have the functionality to rank a column based on value. For plotting tasks, Dask, Anaconda (Matplotlib) and R have the built-in support for plotting.
- **Distribution and Inferential Statistics**: All the systems are able to calculate correlation. However, when it comes to estimating the PDF (probability density function), hypothesis testing and skewness some of the systems do not have the functionality. The PDF is calculated using the kernel density estimator. The PostgreSQL (MADlib) does not have any available functionality for PDF. Dask has a rolling skew function, but the max window size is the size of a partition, which is smaller than one million. In our experiments, PySpark did not support efficient hypothesis testing for datasets with over one million rows.
- **Time Series**: Time series is a sequence of data points indexed by time order. The two tasks in this suite are autocorrelation and EWMA (exponential weighted moving average). Anaconda, Dask and R have built-in functionality to calculate autocorrelation. PostgreSQL or MADlib does not have either feature built-in and we have not found an efficient user defined function (UDF). Dask has built-in functionality for autocorrelation, but not for EWMA. PySpark does not have built-in functionality for either of the time series tasks.
- **Machine Learning**: The machine learning tasks are: linear regression, K-means Clustering and Naive Bayes Classification. All the the systems or libraries used by the systems have built-in functionality to complete these tasks. Dask and Anaconda use the same library scikit-learn, PostgreSQL use MADlib, whereas, R and PySpark have built-in functionality in the library.

2) Data model for the micro benchmark: We created a synthetic dataset generator that can generate data tables (as text files) with different scale factors. As shown in Table III, each dataset consists two data tables: data1 and data2. For a given scale factor, both the data tables contain the same number of rows. For instance, with scale factor (SF) 1, each data table has 1 million rows and with SF 100, each has 100 million rows. Three different scale factors were used for the scalability experiments: SF 1, 10, and 100.

The schema and the number of columns of data1 and data2 differ. Table data1 contains one time series, two string, two integer and three float columns. The time series column is sequential and starts at the year 1970 and the time interval between each row shrinks as the scale factor grows. One string column values are chosen uniformly from a list of 1,000 elements and the other string column values are chosen from a Zipf distribution of a list of 100,000 elements. One integer column values are randomly chosen between 0 and 1,000,000 and the other between 0 and 2^{31}. The values of the three float

Operation	TaskId	Description	Support in the evaluated systems				
			Anaconda	Dask	PostgreSQL	PySpark	R
Basic File I/O							
Read	read	Read from data1	Y	Y	Y	Y	Y
Write	write	Write data1 to a csv file	Y	Y	Y	Y	Y
Data Wrangling							
Sorting	sort	Sort data based on the column uni in data1	Y	Y	Y	Y	Y
Filtering	filter	Filter the column rand1 of data1, if smaller than a given value	Y	Y	Y	Y	Y
Merging	merge	Merge on column rand1 from both data1 and data2	Y	Y	Y	Y	Y
Group By	groupby	Group by column city in data1	Y	Y	Y	Y	Y
Removing Duplicates	remdup	Remove duplicates from column words in data1	Y	Y	Y	Y	Y
Descriptive Statistics							
Central Tendency	centend	Find the mean of uni, mode of rand1 and median of rand2 in data1	Y	Y	Y	Y	Y
Measure of dispersion	dispers	Finds the range of exp and standard deviation of uni column in data1	Y	Y	Y	Y	Y
Rank	rank	Add a new column based on rank of column uni in data1	Y	NF	Y	Y	Y
Outliers	outlier	Remove the outliers of column exp in data1	Y	Y	Y	Y	Y
Scatter Plot	scatter	Draw the scatterplot based on column uni and nor in data1	Y	Y	NF	NF	Y
Distribution and Inferential Statistics							
Probability Density Function (PDF)	pdf	Find the Pdf of the data series on column rand1	Y	Y	NF	Y	Y
Skewness	skew	Calculate the skewness of the data1 series on column rand2	Y	NF	Y	Y	Y
Correlation	corr	Calculate the correlation between uni and exp columns in data1	Y	Y	Y	Y	Y
Hypothesis Testing	hypo	Hypothesis testing (shuffling method)	Y	Y	Y	NF	Y
Time Series							
Exponentially-weighted moving average (EWMA)	ewma	Find the exponentially-weighted moving average on column rand1	Y	NF	NF	NF	Y
Autocorrelation	autocorr	Find the autocorrelation of the nor column in data1 time-series	Y	Y	NF	NF	Y
Machine Learning							
Linear Regression	linreg	Linear Regression between columns rand1 and rand2 in data1	Y	Y	Y	Y	Y
Clustering	kmeans	K-means Clustering with two clusters between columns rand1 and rand2	Y	Y	Y	Y	Y
Classification	naivebayes	Naive Bayes (rand1 as the target and rand2 and uni in data1 as features)	Y	Y	Y	Y	Y

TABLE II
MICRO BENCHMARK TASKS AND FEATURE MATRIX (NF=NO BUILT-IN FUNCTIONALITY)

columns are selected from a normal distribution, an exponential distribution and a uniform distribution respectively.

Table data2 contains one string, one integer and two float columns. The string column values are chosen from a uniform distribution from a list of 1000 strings, the integer column values are randomly distributed between 0 to 1,000,000 and the float columns values are selected from a normal distribution and a uniform distribution between 0 and 2^{31}.

Data table	**Scale factor**					
	1		**10**		**100**	
	Rows (Mil.)	Size (MB)	Rows (Mil.)	Size (MB)	Rows (Mil.)	Size (GB)
data1	1	92.6	10	926.3	100	9.3
data2	1	43.9	10	438.9	100	4.4

TABLE III
MICRO BENCHMARK DATASET: CARDINALITY FOR VARIOUS SCALING FACTORS (MIL.=MILLION)

Data table	**Rows (K)**	**Size (MB)**	**Benchmark application**
weather	35,064	0.863	Smart Grid Analytics - Demand Forecast
consumption	35,064	0.875	Smart Grid Analytics - Demand Forecast
hockey_stats	888	1.2	Sports Analytics - Corsi in Ice Hockey

TABLE IV
MACRO BENCHMARK DATASET (K=THOUSAND)

B. Macro benchmark

With the rapidly rising volume of data, the importance of data science is growing in many application domains. The goal of our macro benchmark is to model some of these applications and assess the performance of the evaluated data systems with each. In our macro benchmark we have included two real-world applications. Each application consists of a series of data science tasks that are executed in sequence. Table V summarizes these applications and which data systems have support for which tasks from these application. In the next section, we briefly describe each application.

1) Macro benchmark applications:

- **Smart Grid Analytics - Demand Prediction**: The demand prediction application focuses on predicting power consumption over a year for a particular substation based on the HDD (heating degree day) and CDD (cooling degree day) curves. The tasks for the demand prediction are: reading data from multiple CSV files, create 5 minute time series intervals from 2013 to 2016, merge all the data, add two separate columns, filter the data based on two values, predict the consumption for the years and correlate the real results with the predicted results.
- **Sport Analytics - Corsi in Ice Hockey**: The hockey analytics application focuses on calculating well-known and used statistics called Corsi and Fenwick. These are used to calculate the player effectiveness while on the ice. The tasks of finding Corsi and Fenwick are reading the data from a CSV, calculate Corsi, as well as Fenwick, based on several statistics, split the players based on position and group the players by team so one could evaluate each team puck possession skills.

2) Data model for macro benchmark: The details of the datasets in the macro benchmark are shown in Table IV. The data used for the smart grid demand prediction came from two sources: historical weather data for New Brunswick from Environment Canada [7], and historical New Brunswick power demand from NB Power [18]. The data used for the sports analytics came from the hockey analytic database [13].

VI. IMPLEMENTATION

In this section we discuss about the benchmark implementation and the challenges we encountered along the way.

A. Implementation overview

The details of the benchmark tasks can be found in the Description columns of Table II and V. The source code of Sanzu can also be inspected to glean additional details. While implementing a task, we tried to use as much built-in functionality as possible. If the built-in functionality was absent, no effort was made to implement it in a given system. However, if open source libraries were available with that functionality, it was implemented. An issue with using the built-in software was the different built-in defaults. For example, while sorting data if there was a tie, each system would arrange the tied values based on a different criteria.

B. Challenges

Anaconda and R were similar in terms of implement effort, as they both have sufficient documentation Dask extends libraries used in Anaconda, but only partially. As a result, implementation in Dask was straightforward until a certain functionality was missing; this resulted in an incomplete task. PySpark and PostgreSQL required more development effort than the other systems. Sometimes, otherwise simple operations needed significant coding effort. For example, in PySpark there are only two ways to get rows by index while using a dataframe. The first is to add a new column for the index manually and filter based on this. The second is to convert to another data-type or structure, perform slicing and indexing operations there, and then convert back to a dataframe. This issue presented itself when developing the hypothesis testing, as there was no efficient way to shuffle (randomize) data and then split it based on the index, into a PySpark dataframe. Furthermore, when working with the macro benchmark (i.e. real-world messy data), cleaning the data up is much more difficult while using PostgreSQL or PySpark, as opposed to Anaconda or R.

While running tasks involving certain aggregate functions (e.g. finding the median and mode), PostgreSQL 9.3 took too long even with dataset scale factor 1. After upgrading PostgreSQL to the latest version (9.6), the issues were resolved.

VII. EXPERIMENTAL SETUP

During the experiments, for each task and for a specific data system we loaded the data first. Then each task was run four times. We exclude the first run from the results as this can be affected by the loading of the data. We report execution times of each task with error bars.

In-house data science tasks are usually run on a single machine or on a small cluster. Since PySpark is a distributed

Operation	TaskId	Description	Support in the evaluated systems				
			Anaconda	Dask	Postgre-SQL	PySpark	R
Smart Grid Analytics - Demand Prediction							
Reading table data	reading	Read data from CSV files	Y	Y	Y	Y	Y
Create data	create	Create five minute time series data	Y	Y	Y	Y	Y
Join tables	merge	Join consumption and weather tables on time columns	Y	Y	Y	Y	Y
Added HDD Heating Degree Day	addcol	Calculated heating degree day from outside temperature	Y	Y	Y	Y	Y
Filter	filter	filter on columns HDD and on the hour of the day	Y	Y	Y	Y	Y
Find curves	2linereg	Use linear regression to find heating and cooling curve from HDD and consumption	Y	Y	Y	Y	Y
Predict consumption	predict	Predict power consumption for 3 years from the curves and outside temperature	Y	Y	Y	Y	Y
Correlation	corr	Use correlation to see if curves predicted realistic results	Y	Y	Y	Y	Y
Sport Analytics - Corsi and Fenwick in Ice Hockey							
Reading data	reading	read the data from a csv	Y	Y	Y	Y	Y
Corsi statistics	corsi	Add multiple columns for each row/player for Corsi	Y	Y	Y	Y	Y
Fenwick statistics	fenwick	Add multiple columns for each row/player for Fenwick	Y	Y	Y	Y	Y
Split players on position	split	Filter the players based on forwards and defense	Y	Y	Y	Y	Y
Group players	groupby	Group payers based on their team	Y	Y	Y	Y	Y

TABLE V
MACRO BENCHMARK TASKS AND FEATURE MATRIX

data processing system, we ran it in a cluster of four machines with standardized software. The other systems were run on a single machine. Each machine has 8 GB memory and 4 Intel i5-2400 CPUs running at 3.10 GHz and ran Ubuntu 14.04 OS.

VIII. Benchmark Results and Discussion

In this section we present our evaluation of the systems with the benchmark. We introduce the micro benchmark results first, and then the macro benchmark results.

A. Micro benchmark results

We conducted two different series of experiments to evaluate the data systems with the micro benchmark task suites. The first series of experiments (Section VIII-A1) were run with dataset scale factor (SF) 1 i.e. 1 million rows per data table and the goal was to determine relative performance and feature support. The aim of the second series of experiments (Section VIII-A2) was to evaluate scalability of the systems with increased data size. So for these we used all three datasets with SF 1, 10 and 100.

1) Performance and functionality: As can be seen in Figure 2, there is no system that is the best across-the-board in all tasks. On the basic file I/O, R is the slowest at reading, however it is competitive in writing, while PySpark is the fastest at both. In the data wrangling suite, Anaconda is consistently the fastest and PostgreSQL is the slowest in all but the group by, and the others vary depending on the task. In descriptive statistics suite, Anaconda is consistently the fastest, R is in second in all but one tasks. PostgreSQL, Dask and PySpark lack some functionalities (shown as 'NA'), and PostgreSQL outperforms Dask and PySpark in every task it has functionality. In the distribution and inferential statistics suite Anaconda and R are the fastest in all cases; Dask, PySpark and PostgreSQL are slower and lack different functionalities for certain tasks. In the time series suite, Anaconda is the fastest. R also has functionality for both time series tasks, but it is significantly slower. In contrast, Dask only supports autocorrelation, and PySpark and PostgreSQL do not have functionality for either task. As for the machine learning task suite, Anaconda is the fastest in two out of the three tasks, while PostgreSQL is the slowest in two out of the three. These results suggest that at dataset scale factor 1, Anaconda's performance is the most consistent and it is the best choice for most tasks. R also supports all the features, but it is slower than Anaconda in all but two tasks. As for Dask, PySpark and PostgreSQL, their performance and functionality depend on specific tasks. Thus at SF 1, whenever the data fits in memory, Anaconda is the best choice for the combination of functionality and performance.

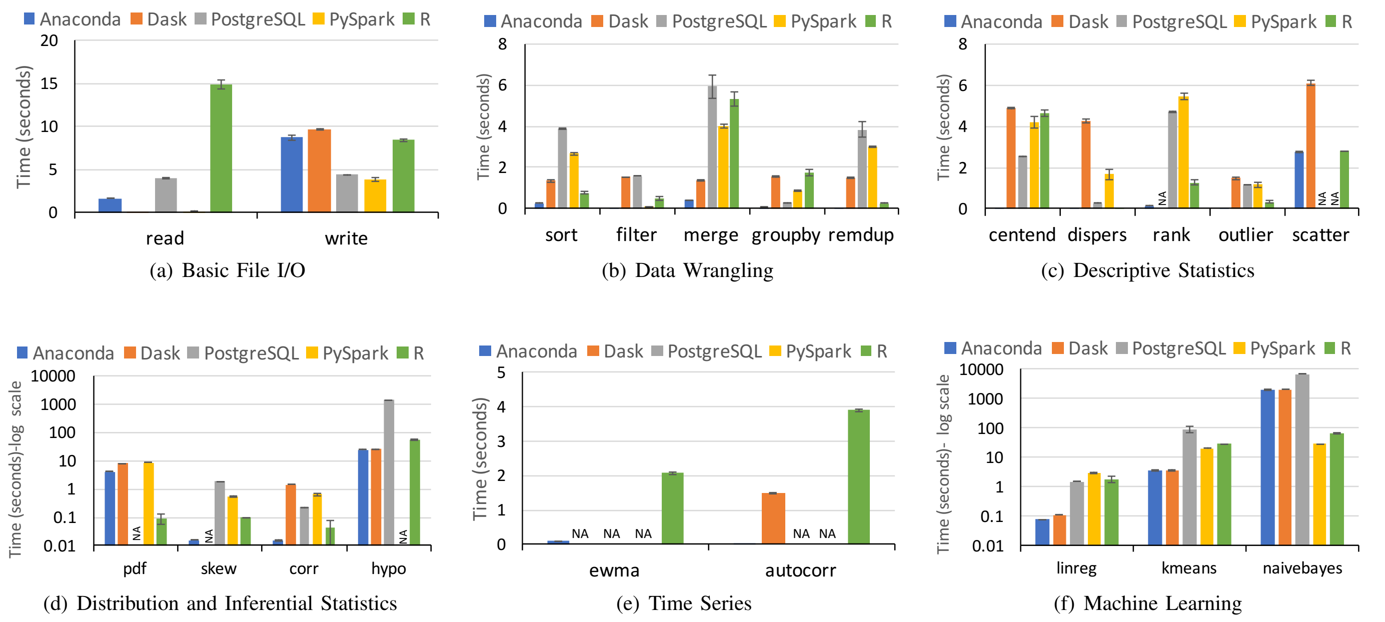


Fig. 2. Execution times of the tasks in the Micro Benchmark suites (scale factor 1: 1 million)

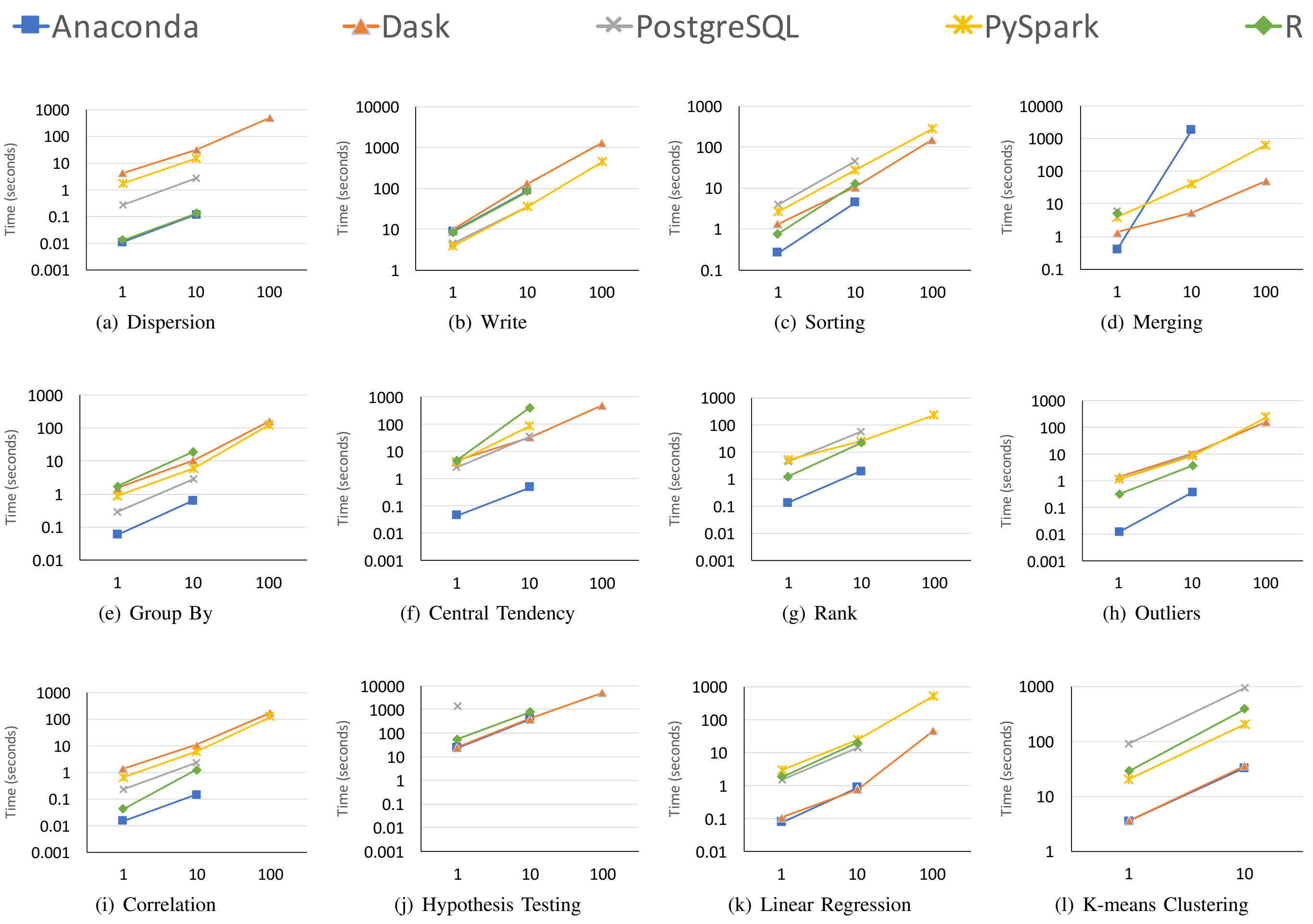
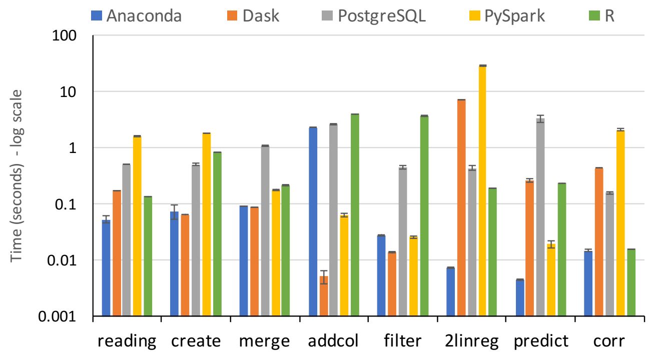


Fig. 3. Scalability of **selected** Micro benchmark tasks with dataset scale factors: 1, 10 and 100 (All y-axis are in log scale)

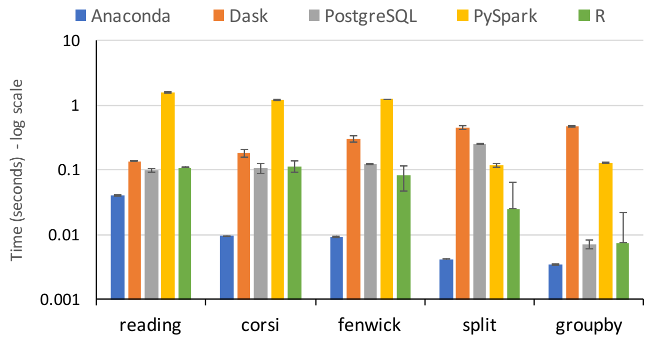
2) Scalability: Table VI summarizes information about scalability of the data systems for each task of all the micro benchmark suites. In Figure 3 the scalability results are presented for some selected tasks.

In almost all cases all the systems scale up to SF 10 (ten million rows per data table). The only exceptions are the merge operation, which joins both data tables. As the memory of the systems start to run out, R and PostgreSQL throw memory errors, whereas Anaconda is significantly slower than Dask and PySpark. Naive Bayes Classification did not scale up on any of the systems except for R, while hypothesis testing scaled for all except for PostgreSQL.

For the SF 100 (100 million rows per data table), the R data loading was taking over two hours to read, PostgreSQL ran out of memory that caused the machine to restart and there was a memory error thrown during the read for Anaconda. As a result, the rest of the tasks for these systems were not evaluated. It is important to note that if the task failed or took too long on a lower row count, it was not evaluated again on a higher SF. For the systems that we were able to run at SF 100, their results mostly stayed consistent with the results at a lower SF. Some tasks, however, did not scale up from SF 10 to 100. For Dask, it was only with K-means Clustering that did not scale. Whereas for PySpark, central tendency, measure of dispersion and K-means Clustering were all unable to scale. These results suggest that in terms of scalability up to SF 100, Dask scales better than others with regards to functionality.



(a) Smart Grid Analytics - Demand Prediction



(b) Sport Analytics - Corsi and Fenwick in Ice Hockey

Fig. 4. Execution times of the tasks in the Macro Benchmark applications

B. Macro benchmark results

In Figure 4 the macro benchmark results are presented. As can be seen, in the Demand Prediction application, there is no single system that is the best across-the-board. Anaconda is consistently competitive and is the quickest in the task reading, linear regression, and prediction. The others vary in competitiveness and performance depending on the task.

In the sport analytics application, Anaconda has the best performance in all of the tasks and R consistently has the second best performance. PostgreSQL is also very competitive with R but is much slower in the split task. PySpark and Dask are consistently slower in overall performance than the others.

C. Ranking the systems

Historically, benchmark results were summarized and presented in several different ways. The authors in [4] discuss how to correctly summarize benchmark results. In [25] geometric mean, normalized to a reference system, was used for ranking. We considered using this approach, but it proved to be challenging. In our benchmark a number of tasks were not supported by some of the systems either due to lack of functionality or because of failure to complete at higher scale factors. Therefore, we decided not to rank the systems.

IX. Conclusions and Future Work

With the rapid growth in data, the need to efficiently analyze data has become paramount. As a result, data science is rising in importance. Data science provides a systematic approach for processing and analyzing data. Although, a number of frameworks and data systems have emerged to support the data science work-flow, there is no standard benchmark to evaluate them. We have presented Sanzu, a benchmark for data science. It includes a micro benchmark to test individual operations and a macro benchmark to represent real-world use cases.

We have presented a performance evaluation of five popular data science frameworks and systems: R, Anaconda Python, Dask, PostgreSQL (MADlib) and PySpark. Our evaluation suggests that some of these systems lack support for specific data science tasks. Furthermore, we have found that several systems do not scale with larger data sets (scale factor 10 or 100). We hope that our findings may lead to improved support for functionality and scalability for the data science tasks within some of the systems.

For our future work, we plan to test the ability of each system to process other data models, by incorporating semi-structured and unstructured datasets. We would like to include additional macro benchmark applications. We would also like to evaluate other data science systems, besides the five systems that we evaluated.

References

[1] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. In *SoCC*, pages 143–154, 2010.

[2] Dask. https://dask.pydata.org/en/latest/.

[3] B. J. Dorr, P. C. Fontana, C. S. Greenberg, M. L. Bras, and M. A. Przybocki. Evaluation-driven research in data science: Leveraging cross-field methodologies. *IEEE Big Data*, pages 2853–2862, 2016.

Operation	**Support in the evaluated systems**														
	Anaconda			**Dask**			**PostgreSQL**			**PySpark**			**R**		
	scale factor			scale factor			scale factor			scale factor			scale factor		
	1	**10**	**100**	**1**	**10**	**100**	**1**	**10**	**100**	**1**	**10**	**100**	**1**	**10**	**100**
Basic File I/O															
Read	Y	Y	F	Y	Y	Y	Y	Y	F	Y	Y	Y	Y	Y	F
Write	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Data Wrangling															
Sort	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Filtering	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Merging	Y	Y	NA	Y	Y	Y	Y	F	NA	Y	Y	Y	Y	F	NA
Grouping	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Removing Duplicates	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Descriptive Statistics															
Central Tendency	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	F	Y	Y	NA
Measure of dispersion	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	F	Y	Y	NA
Rank	Y	Y	NA	NF			Y	Y	NA	Y	Y	Y	Y	Y	NA
Outliers	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Scatter Plot	Y	Y	NA	Y	Y	Y	NF			NF			Y	Y	NA
Distribution and Inferential Statistics															
PDF	Y	Y	NA	Y	Y	Y	NF			Y	F	NA	Y	Y	NA
Skewness	Y	Y	NA	NF			Y	Y	NA	Y	Y	Y	Y	Y	NA
Correlation	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Hypothesis Testing	Y	Y	NA	Y	Y	Y	Y	L	NA	NF			Y	Y	NA
Time Series															
EWMA	Y	Y	NA	NF			L	NA	NA	NF			Y	Y	NA
Autocorrelation	Y	Y	NA	Y	Y	Y	NF			NF			Y	Y	NA
Machine Learning															
Linear Regression	Y	Y	NA	Y	Y	Y	Y	Y	NA	Y	Y	Y	Y	Y	NA
Clustering	Y	Y	NA	Y	Y	F	Y	Y	NA	Y	Y	F	Y	Y	NA
Classification	Y	L	NA	Y	L	NA	Y	F	NA	Y	F	NA	Y	Y	NA

TABLE VI
SCALABILITY OF THE MICRO BENCHMARK TASKS WITH DATASET SCALE FACTORS: 1, 10 AND 100 [NA=NOT APPLICABLE (THE DATASET COULD NOT BE LOADED), NF = NO BUILT IN FUNCTIONALITY AVAILABLE, F=FAILED (OUT OF MEMORY ERROR OR CRASHED THE MACHINE), L=TOOK TOO LONG]

[4] P. J. Fleming and J. J. Wallace. How not to lie with statistics: The correct way to summarize benchmark results. *Commun. ACM*, 1986.

[5] A. Ghazal, T. Ivanov, P. Kostamaa, A. Crolotte, R. Voong, M. Al-Kateb, W. Ghazal, and R. V. Zicari. BigBench V2: The New and Improved BigBench. In *ICDE*, pages 1225–1236, 2017.

[6] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen. Bigbench: Towards an industry standard benchmark for big data analytics. In *SIGMOD*, pages 1197–1208, 2013.

[7] Historical Climate Data. http://climate.weather.gc.ca.

[8] Graysort benchmark. http://sortbenchmark.org.

[9] J. Grus. *Data Science from Scratch*. O'Reilly Media, Inc., 2015.

[10] Apache Hadoop. http://hadoop.apache.org/.

[11] J. M. Hellerstein, C. Ré, F. Schoppmann, D. Z. Wang, E. Fratkin, A. Gorajek, K. S. Ng, C. Welton, X. Feng, K. Li, and A. Kumar. The MADlib Analytics Library: Or MAD Skills, the SQL. *Proc. VLDB Endow.*, 5(12):1700–1711, Aug. 2012.

[12] T. Hey, S. Tansley, and K. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. October 2009.

[13] Hockey stats. http://www.hockeyabstract.com.

[14] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[15] X. Liu, L. Golab, W. Golab, I. F. Ilyas, and S. Jin. Smart meter data analytics: Systems, algorithms, and benchmarking. *ACM Trans. Database Syst.*, 42(1):2:1–2:39, Nov. 2016.

[16] Matplotlib. https://matplotlib.org/.

[17] P. Mehta, S. Dorkenwald, D. Zhao, T. Kaftan, A. Cheung, M. Balazinska, A. Rokem, A. Connolly, J. Vanderplas, and Y. AlSayyad. Comparative evaluation of big-data systems on scientific image analytics workloads. *Proc. VLDB Endow.*, pages 1226–1237, 2017.

[18] NB Power. https://tso.nbpower.com/Public/en/system_information_archive.aspx.

[19] Numpy, scientific computing with Python. http://www.numpy.org/.

[20] NYTimes article. https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html.

[21] Pandas Python Data Analysis Library. http://pandas.pydata.org/index.html.

[22] PostgreSQL. http://www.postgresql.org/.

[23] Anaconda. https://www.continuum.io/.

[24] The R Project for Statistical Computing. https://www.r-project.org/.

[25] S. Ray, B. Simion, and A. D. Brown. Jackpine: A benchmark to evaluate spatial database performance. In *ICDE*, pages 1139–1150, 2011.

[26] R. Schutt and C. O'Neil. *Doing Data Science: Straight Talk from the Frontline*. O'Reilly Media, Inc., 2013.

[27] scikit-learn Machine Learning in Python. http://scikit-learn.org/stable/.

[28] Apache Spark. https://spark.apache.org/.

[29] R. Taft, M. Vartak, N. R. Satish, N. Sundaram, S. Madden, and M. Stonebraker. Genbase: A complex analytics genomics benchmark. In *SIGMOD*, pages 177–188, 2014.

[30] The Transaction Processing Performance Council. http://www.tpc.org.