Leveraging modern big data stack for swift development of insights into social developments

He Huang[1,4] ⋆, Yixin He[3,4], Longpeng Zhang[3], Zhicheng Zeng[4], Tu Ouyang[2], and Zhimin Zeng[4]

[1] University of Melbourne,
Parkville VIC 3010, Australia
hhhu@student.unimelb.edu.au
[2] Computer and Data Science department, Case Western Reserve University
10900 Euclid Ave., Cleveland, OH, 44106, USA.
tu.ouyang@case.edu
[3] University of Electronic Science and Technology of China, ChengDu, China
yixinhe09@std.uestc.edu.cn
zlp1988@uestc.edu.cn
[4] Zilian Tech Inc., ShenZhen, China
zengzc@ziliantech.net
zengzm@tsingzhi.cn

Abstract. Insights of social development, presented in various forms, such as metrics, figures, text summaries, whose purpose is to summarize, explain, and predict the situations and trends of society, is extremely useful to guide organizations and individuals to better realize their own objectives in accordance with the whole society. Deriving these insights accurately and swiftly has become an interest for a range of organizations, including agencies governing districts, city even the whole country, they use these insights to inform policy-makings. Business investors who peak into statistical numbers for estimating current economical situations and future trends. Even for individuals, they could look at some of these insights to better align themselves with macroscopical social trends.
There are many challenges to develop these insights in a data-driven approach. First, required data come from a large number of heterogeneous sources in a variety of formats. One single source's data could be in the size of hundreds of Gigabytes to several TeraBytes, ingesting and governing such huge amount of data is not a small challenge. Second, many complex insights are derived by domain human experts in a trail-and-error fashion, while interacting with data with the aid of computer algorithms. To quickly experiment various algorithms, it asks for software capabilities for infusing human experts and machine intelligence together, this is challenging but critical for success.
By designing and implementing a flexible big data stack that could bring in a variety of data components. We address some of the challenges to infuse data, computer algorithm and human together in Zilian Tech company [20]. In this paper we present the architecture of our data stack and articulate some of the important technical choices when building such stack. The stack is designed to be equipped with scalable storage that could scale up to PetaBytes, as well as elastic distributed compute engine with parallel computing algorithms. With these features the data stack enables *a*) swift data analysis, by human analysts interacting with data and machine algorithms via software support, with on-demand question answering time reduced from days to minutes; *b*) agile building of data products for end users to interact with, in weeks if not days from months.

Keywords: cloud, data stack, social development

1 Introduction

The potential benefits are immense by drawing on large-scale online and commercial data to construct insights of social development, for example, trends in economic and business development, emerging patterns of people's daily life choices, comparative technology advances of competing regions, population sentiment to social events and so on. These insights are valuable, sometimes critical, in scenarios like helping government agencies for more objective policy making, aiding decision-making of investors before pulling money into certain business in certain regions, even helping individuals who might just want to check cities and companies' outlooks before settling among several job offers.

⋆ First two authors contribute equally, their work were done when authors interned in Zilian Tech

Recent years have seen many articles to investigate various aspects of social activities and developments based on data and models. Bonaventura et al. [23] construct a worldwide professional network of start-ups. The time-varying network connects start-ups which share one or more individuals who have played a professional role. Authors suggest such network has predictive power to assess potential of early stage companies. [26] investigates foreign interference found on twitter, during the 2020 US presidential election. Natural language processing models are used to classify troll accounts, network flow statistics are leveraged to reveal super-connectors. Drawn on top of analysis results drawn from these models, this report is able to quantify prevalence of troll and super-connector accounts in various politics-inclined communities and these accounts' influence among these communities. Jia et al. [24] devise a risk model of covid-19 based on aggregate population flow data, the model is to forecast the distribution of confirmed cases, identify high risk regions threatened by virus transmission, one such model is built and verified using major carrier data of mobile phone geolocations from individuals leaving or transiting through Wuhan between 1 January and 24 January 2020. Authors suggests the methodology can be used by policy-makers in any nations to build similar models for risk assessment.

To realize many of aforementioned applications, a large amount of data need to be acquired, stored and processed, a scalable and efficient big data processing platform is the key. In our company, we have built such a data platform. We argue that the data stack of our platform provides enough flexibility to incorporate a variety of modern data component implementations and products from different vendors and bring them together to enable data applications to solve our use cases. Mainly two categories of applications are enabled by the design of the data stack: analytics-oriented applications and real-time transactional applications (usually customer-facing). These two application categories suite different use cases when developing data applications for extracting insights of social development. We showcase two concrete applications: one is a notebook-like analytics tool for analysts to examine research publications of a country with the world's biggest population. The other is a customer-facing search application one of whose function is to retrieve and summarize companies' patent statistics in past 20 years of the same big country.

This paper's main contributions are not on advancing techniques of individual data components, but more of a practical study on how to incorporate appropriate data techniques under a flexible stack framework we propose, to enable real-world data-oriented user cases with minimum time-to-market. We document technical trade-offs we made for choosing the right set of components and technologies, from many existing ones, we use these components to compose a cohesive platform that suits our use cases.

In the following of this paper, section 2 presents the architecture of the big data stack, then dive into the technical reasoning to choose concrete techniques for several key components. Section 3 shows two example applications and explain how the big data stack enable swift development, followed by the conclusion in Section 4.

2 The Big Data Stack

Figure 1 depicts a high-level view of what are in the big data stack, the stack is composed of five key components. In the past decade, we have seen a blossom of technologies that could possibly be used to implement the components of proposed stack. Too many techniques sometimes bring no help, but on the contrary quite a lot challenges for a system architect, who need to carefully compare and make trade-offs between several technologies and eventually decide on the right one to have it incorporated into one single cohesive stack.

The applications that we want the techniques to enable are mainly two categories: analytics-oriented and real-time customer facing. To enable these two categories, we set out with a number of goals for choosing the techniques to implement the data stack. First major goal is *flexibility*, we strive to the keep our options open to be able to switch to a different technique in the future in needed and avoid being locked into certain set of techniques. *Scalability* and *agility* are two goals for analytics-oriented applications. *Responsiveness* is one goal for real-time applications, "real-time" means the processing time is within the order of sub-second.

Below we dive into technical reasoning in each component of our stack, about the choices of concrete techniques. Note that, the index numbers of the list items correspond to the labels of components in Figure 1.

1. **Data Governance**
   The social development data could come as structured, e.g., files with clearly defined schema, e.g., CSV, parquet [28] files; or semi-structured, like XML and JSON; or unstructured, e.g., pictures, audio files, videos. The existing and emerging storage technologies to choose include: structured-data-only traditional database, that aims to store key operational data only; data warehouse that are designed to stores all your data but mainly structured data, snowflake [17] and Oracle [10] are examples of such warehouse providers. Recent data lake technologies [25],

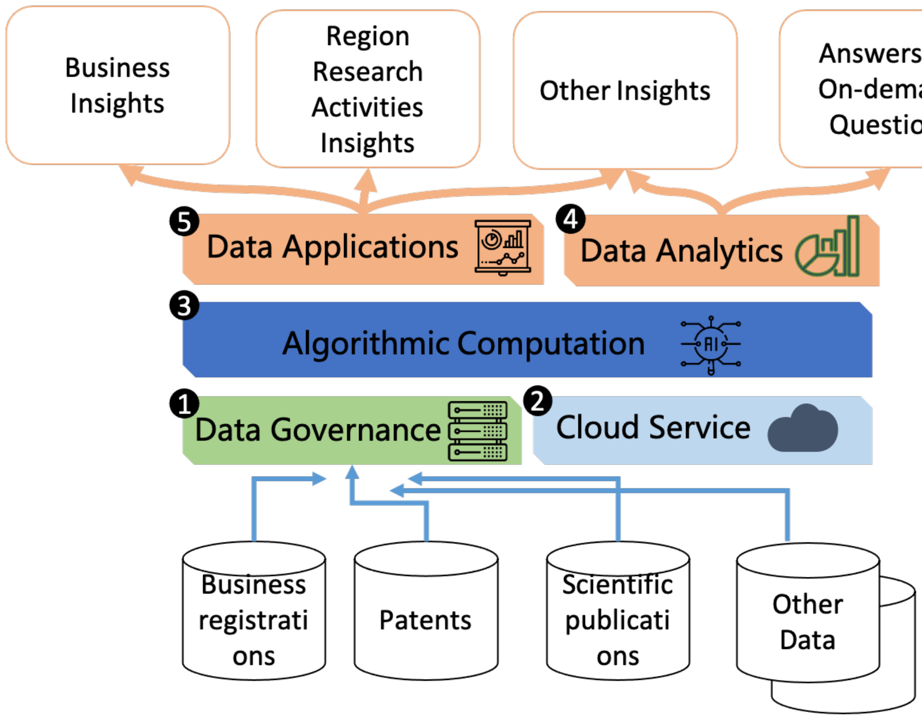


Fig. 1: The conceptual view of the big data stack of our data intelligence platform
Each component in this stack figure has a corresponding text paragraphs of the same label for more detail.

that promise to be able to store huge amount of structured and unstructured data. Data lakehouse [21] is another recent data storage paradigm attempting to unify both data warehouse and data lake. We keep an open mind in choosing storage technologies since we believe at this time not a single existing technology mature enough to solve all the cases. When picking technologies for our stack, we decide on data lake storage techniques for raw data storage for analytics-oriented applications that meet the goal of *scalability*. While for real-time applications, we integrate traditional relational databases for its optimized transaction handling for *responsiveness*.

2. **Cloud Service**
   Fifteen years after the launch of AWS, we now enjoy a competitive cloud service provider market. There are global leading providers like AWS [2] and Azure [8], as well as region challengers like AliCloud [1], OVHCloud [11], the cloud services offered by different providers are more or less overlapped and converged gradually. The choice of providers sometimes more rely on business factors, like the availability of that providers in the region of target markets. We build internal software tools to abstract away the native cloud services from our applications as much as possible, we invest on Kubernetes technologies [27] as the application runtime environment so that we keep the option open to later evolve the stack for hybrid or multi-clouds if needed. Using cloud service enables *scalability* both in storage and computation.
3. **Algorithmic Computation**
   Distributed data computation engine that provides parallel-processing capabilities is key to analytics-oriented applications processing massive datasets. Spark [4] and Flink [3] are two leading techniques. Flink is from the beginning a streaming-oriented data processing engine while Spark is more popular engines for batch processing and is catching up in streaming. We choose Spark as the our stack's compute engine, because we consider Spark is better positioned in the whole data processing ecosystem. Many technologies come with existing solutions to integrate with Spark, with that we could enjoy more flexibility on choosing other techniques and know they will integrate well with the compute engine. This computation component is related to, and interleaved with the data analytics component described below.
4. **Data Analytics**
   Many open source tools to choose from for data analysis, tools used in single machine include Pandas [12], Scipy, sklearn [15]. We prioritize to support tools in the stack that are able to run on multiple machines in order to harvest distributed computing power provided by the cloud, to support *agility* for analytics-oriented applications. Spark

is our chosen technique that provides the desirable distributed computation capability, additionally Spark provides APIs in SQL semantic that is familiar to many data-analysis specialists already.

Tensorflow [18] and PyTorch [13] are two machine learning tools that we aim to integrate into our platform.

The design principle in this data analytics component is not to lose the flexibility and being able to integrate more tools in the future if necessary. We try to best to avoid locking into a handful of tools pre-maturely. Tools that have low learning curves are preferred, because *agility* is one main goal. We try to reduce as much as possible the unnecessary effort of an analyst to wrestle with unfamiliar tooling concepts or APIs.

5. **Data Applications**

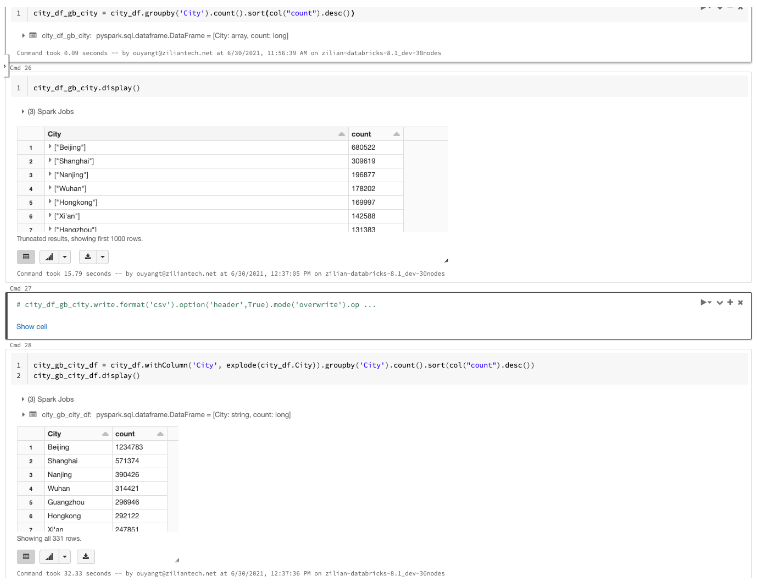
   We leverage open-source frontend Jupyter [7] to build analytics-oriented applications. We also use data visualization tools directly from some cloud vendors, e.g., PowerBI from Azure. When choosing such a specific data visualization tool from one vendor, we usually examine whether it supports many data input/output techniques rather than only those from the same vendor. We decide on frontend frameworks such as Vue and ReactJS [19, 14], and backend frameworks such as NodeJS and Django [9, 6], to build customer-facing real-time applications. These techniques have matured, they have been integrated and tested in cloud environments for many years. In addition there are existing open source data connectors for the frameworks we choose, for connecting them to different data storage techniques so that we keep the *flexibility* and not being locked into certain techniques. Another principle we have is to bias the choices on those that we could quickly prototype with, and then iterate on the prototype with fast turn-around time, this helps to achieve our *agility* goal.

3 Two Example Applications Enabled By the Stack

In this section, we showcase two example applications built on top of our data stack.

One analytics-oriented application shown in Figure 2a is to investigate academic paper publication trends in each major city of China, for assessing cities' research activity levels. The research publication data we collected contains around 8 millions entries, organized as JSON files in ~ 80 Gigabytes. We use one cluster that consists of 30 nodes, each node of which has 14GB memory and 8 CPU cores, for data processing. Spark, the compute engine running on this cluster, orchestrates distributed computation tasks of analysis code. We choose a browser-based Jupyter [7] notebook environment for analyst to program analysis code, analysis results returned by the compute engine are also shown on the same UI. The programing API is a combination of SQL and DataFrame [22], both are familiar to experienced analysts, in fact our analysts put these new tools in use in a matter of a few hours' learning. Figure 2a shows the UI of this browser-based programming tool for analyst's use, backed by a powerful distributed cluster underneath. After loading the data into the cluster memory within minutes, analyst could use family APIs to program and then execute analytics tasks on the cluster. One example task is to group the publications by individual cities, then sort the cities by the publication numbers this particular analysis task takes less than one minute on the whole 80GB dataset. With swift turn-around time of many such analytics tasks, analysts feel enabled and motivated to explore more analysis questions and experiment more analysis approaches to solve same questions.

Another application depicted in Figure 2b is a customer-facing information-search web application. This application provides a search function for companies with their patent statistics in different regions of China. We leverage a cluster of 60 nodes, each has 14GB memory and 4 CPU cores, for running routine batch jobs to calculate key metrics that power the search. One of most expensive task in these routine batch jobs is to calculate region-specific company patent metrics, which needs to perform an expensive *SQL JOIN* of two large datasets: one is company registration data of past 20 years, consisting of $\sim$ 36MM entries the other dataset is patent data of past 20 years that includes $\sim$ 28MM patent entries. First a *SQL JOIN* of these two large dataset and then a *SQL GROUPBY* to group companies with patents by different regions. In total this task takes around 12 minutes by Spark engine on this cluster. The resulting metrics are then inserted into a PostgrepSQL relational database, which in turns powers the web search application. The search portal responses to users' search with results in few seconds. Figure 2b shows on such search result page, a geographical map of all regions is on the left side, where each region is colored according to its magnitude of numbers of company that have patents, on the right side is the top 10 regions. We are able to build this data application, from ingesting raw data, to setting up batch jobs for analysis, then eventually having web search application powered by a relational database, in weeks. The cohesive data stack connects a number of data storage and compute technologies together, enabling this swift development.



```
city_df_gb_city = city_df.groupby('City').count().sort(col("count").desc())
```

```
city_df_gb_city:  pyspark.sql.dataframe.DataFrame = [City: array, count: long]
Command took 0.09 seconds -- by ouyangt@ziliantech.net at 6/30/2021, 11:56:39 AM on zilian-databricks-8.1_dev-30nodes
```

Cmd 26

```
city_df_gb_city.display()
```

```
(3) Spark Jobs

     City                 count
1    ["Beijing"]          680522
2    ["Shanghai"]         309619
3    ["Nanjing"]          196877
4    ["Wuhan"]            178202
5    ["Hongkong"]         169997
6    ["Xi'an"]            142588
7    ["Hangzhou"]         131383
Truncated results, showing first 1000 rows.

Command took 15.79 seconds -- by ouyangt@ziliantech.net at 6/30/2021, 12:37:05 PM on zilian-databricks-8.1_dev-30nodes
```

Cmd 27

```
# city_df_gb_city.write.format('csv').option('header',True).mode('overwrite').op ...
```

Show cell

Cmd 28

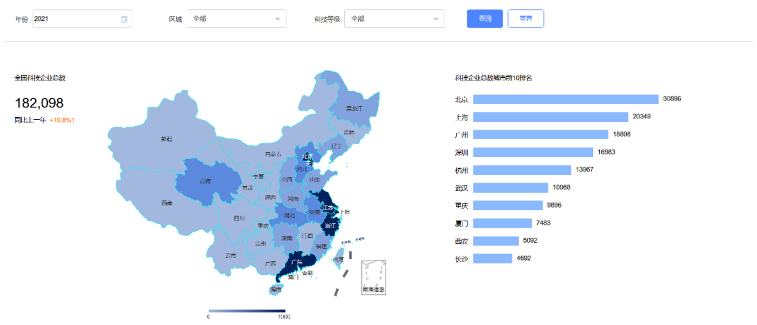
```
city_gb_city_df = city_df.withColumn('City', explode(city_df.City)).groupby('City').count().sort(col("count").desc())
city_gb_city_df.display()
```

```
(3) Spark Jobs
city_gb_city_df:  pyspark.sql.dataframe.DataFrame = [City: string, count: long]

     City         count
1    Beijing      1234783
2    Shanghai     571374
3    Nanjing      390426
4    Wuhan        314421
5    Guangzhou    296946
6    Hongkong     292122
7    Xi'an        247851
Showing all 331 rows.

Command took 32.33 seconds -- by ouyangt@ziliantech.net at 6/30/2021, 12:37:36 PM on zilian-databricks-8.1_dev-30nodes
```

(a) The notebook UI to enable analyst to develop analytics programs quickly



(b) A search result page shows numbers of companies with patents in different regions

Fig. 2: Two applications built on top of the big data stack

4 Conclusion

We present a design of big data stack that collectively function as data intelligence platform, for swiftly deriving social development insights from huge amount of data. We present the concrete techniques to implement this stack, as well as the underlying reasonings on why choosing them among many other choices. The two showcases exemplify two categories of applications this data stack enables: analytics-oriented applications and real-time applications.

We hope to spur discussions on related topics in the community that would also benefit future development of our stack. The better the stack, the better it serves the purpose of providing insights and intelligence to aid informed decision-making of the society.

For future developments, one direction we are looking at is data-mesh like architectural paradigm [16, 5], the purpose is to unlock access to a growing number of domain-specific datasets located within different organizations. Another direction is to ingest and process streaming data in near real-time. For example, extracting information real-time news feed. We consider this a great technical challenge to our data stack and we need to bring in new techniques carefully. Should it be implemented in our data stack, many interesting applications became feasible. We believe the impact, particularly to present decision-makers with near real-time insights from data, would be huge.

5 Acknowledgments

This work is partially supported by National Social Science Foundation of China (Grant No. 20CJY009).

Bibliography

[1] Alibaba cloud services. `https://www.aliyun.com`.

[2] Amazon web services (aws) - cloud computing services. `https://aws.amazon.com`.

[3] Apache flink: Stateful computations over data streams. `https://flink.apache.org/`.

[4] Apache spark - unified analytics engine for big data. `https://spark.apache.org/`.

[5] Data mesh principles and logical architecture. `https://martinfowler.com/articles/data-mesh-principles.html`.

[6] Django: The web framework for perfectionists with deadlines. `https://www.djangoproject.com/`.

[7] Jupyter notebook. `https://jupyter.org/`.

[8] Microsoft azure: Cloud computing services. `https://azure.microsoft.com`.

[9] Node.js. `https://nodejs.org`.

[10] Oracle data warehouse. `https://www.oracle.com/database/technologies/datawarehouse-bigdata.html/`.

[11] Ovhcloud. `www.ovh.com`.

[12] pandas - python data analysis library. `https://pandas.pydata.org/`.

[13] Pytorch - an open source machine learning framework. `https://pytorch.org/`.

[14] React – a javascript library for building user interfaces. `https://reactjs.org/`.

[15] scikit-learn. `https://scikit-learn.org`.

[16] Service mesh. `https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh`.

[17] Snowfalke, data cloud. `https://www.snowflake.com/`.

[18] Tensoflow - an end-to-end open source machine learning platform. `https://www.tensorflow.org/`.

[19] Vue js framework. `https://vuejs.org`.

[20] Zilian tech, shenzhen, china. `http://tsingzhi.cn/About-Us/`.

[21] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. CIDR, 2021.

[22] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, et al. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1383–1394, 2015.

[23] M. Bonaventura, V. Ciotti, P. Panzarasa, S. Liverani, L. Lacasa, and V. Latora. Predicting success in the worldwide start-up network. *Scientific reports*, 10(1):1–6, 2020.

[24] J. S. Jia, X. Lu, Y. Yuan, G. Xu, J. Jia, and N. A. Christakis. Population flow drives spatio-temporal distribution of covid-19 in china. *Nature*, 582(7812):389–394, 2020.

[25] P. P. Khine and Z. S. Wang. Data lake: a new ideology in big data era. In *ITM web of conferences*, volume 17, page 03025. EDP Sciences, 2018.

[26] W. Marcellino, C. Johnson, M. N. Posard, and T. C. Helmus. Foreign interference in the 2020 election: Tools for detecting online election interference. Technical report, RAND CORP SANTA MONICA CA SANTA MONICA United States, 2020.

[27] G. Sayfan. *Mastering kubernetes*. Packt Publishing Ltd, 2017.

[28] D. Vohra. Apache parquet. In *Practical Hadoop Ecosystem*, pages 325–335. Springer, 2016.