Big Data Management System: A Computer View

Yao Wu, Tao Huang, Yuncheng Wu, Hong Chen, Cuiping Li
Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, Beijing, China
School of Information, Renmin University of China, Beijing, China
Email: {ideamaxwu, taohuang, yunchengwu, hongchen, cuipingli}@ruc.edu.cn

***Abstract*—Big data is emerging and rumors go around. Big data is changing the way people work, live and leisure. Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, data curation, search, sharing, storage, transfer, visualization, querying and information privacy. In this paper, we propose datar, a new prospective of BDMS from the point the architecture leveraging ideas from computer. We introduce the five key components of datar by reviewing the the current status of big data management system. We also present evaluation and application of datar and conclude with future directions. We review researches on BDMS and discuss open problems and give poteantial frontier by examing its systematic architecture as datar.**

I. Introduction

A. From Computer to Datar

As Turing proposed the question "Can machines think?" [1], the imitation game begins. However, before that, Von Neumann had started the engineer research on computer and described the logical design of a computer using the stored-program concept in 1945, which has controversially come to be known as the von Neumann architecture [2]. More ago, English mathematician and computer pioneer Charles Babbage proposed the Analytical Engine, a mechanical general-purpose computer designed. The goal of the pioneers is to design a computing machine better than human brain, which can liberate the tedious computing from manual work.

During the last several decades, data management principles such as physical and logical independence, declarative querying and cost-based optimization have led to a multi-billion dollar industry. More importantly, these technical advances have enabled the first round of business intelligence applications and laid the foundation for managing and analyzing Big Data today. The many novel challenges and opportunities associated with Big Data necessitate rethinking many aspects of these data management platforms, while retaining other desirable aspects. The practice [3] and theory [4] contribution of Bachman and Codd open up the research on database. And the steps never stop, Ingres [5], Postgres [6], Mariposa [7], C-Store [8] and VoltDB [9].

Since the computing power of machines becomes stronger, we can notice the shift to data management to explore more in-sight information and knowledge from data. From the computer to datar, we never stop liberating human from tedious and complex work. Although BDMS is a complex set of functionality, we think is as an amplified datar. In point of this view, we summarise and describe the main components in big data management system, to provide a full understanding for beginners and experts.

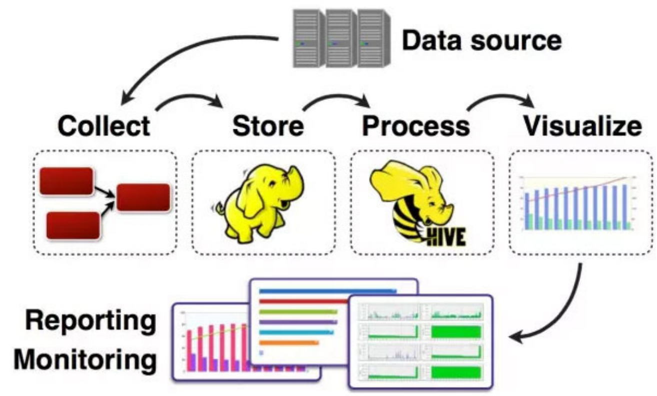


Fig. 1. A typical workflow for big data.

As shown in Fig. 1, BDMS consists of several core components such as, collect, store, process and visualize. Compared with traditional DB systems, BDMS architecture is more flexible and open due to various focus-ons. In this paper, we unify the BDMS as **datar**, a proposed general architecture to design and build BDMS, with respect to the term **computer**.

As we all knnow, the popular computer structure is divided into five parts, input, storage, compute, control and output, in which, computation is the center. If you look closely, you will find that, the BDMS is just the same as computer, consisting of input (data feed), storage, compute (data analysis), control (transaction) and output (visualization), in which, storage is the center. In other words, we can call a computer Fast Computation Processing System (FCPS). Likewise, the BDMS is called as datar, a big data management system centered at data. To better explain the similarity and differences between a computer and a datar, in terms of architecture, we use Fig. 2 to illustrate. In Fig. 2 (a), five core components of a computer are shown in separate rectangles, while in Fig. 2 (b), five corresponding parts are shown. A computer and a datar share the similar functionalities with different emphases on computation and data.

B. What is Datar

Definition (Datar) *A datar is a stack of coherent softewares with specific functionality that can be applied to mine valuable information from persisted data automatically, where specific functionality refer to input, storage, compute, control and output of the big data.* In this paper, we implement datar based on AsterixDB [10] with its extensions as an materilization of this envisioned architecture. This is an data storage centered solution for datar implementation.

A datar, i.e., full-function BDMS, consists of five parts, data feed, data storage, data analysis, data transaction and data visualization. Compared to the computation-centered computer, a datar is data-centered. We take AsterixDB for example, which is a new, full-function BDMS. Data feed is how the data gets into the system. In AsterixDB, data feeds are a built-in mechanism allowing new data to be continuously ingested into system from external sources, incrementally populating the datasets and their associated indexes [11]. Data storage is how the data stored in the system and how the index are built. In AstrixDB, data and index are stored based on LSM structure [12]. Data analysis is how to mine valuable information from stored data. A bunch of methods can be applied, such as popular in-memory computation framework, Spark. Besides, the excution of data processing is also part of data analysis, like Hyracks [13] in AsterixDB. Data transaction is how to control data when it is processed. It is different from the traditional DB systems which have strict ACID. Another very important aspect of datar is visualization. Cloudberry[1] is a research prototype to support interactive analytics and visualization of large amounts of spatial-temporal data using AsterixDB.

C. How Far Datar Can Go

In the late 1970s, the concept of "database machine" emerged, which is a technology specially used for storing and analyzing data. With the increase of data volume, the storage and processing capacity of a single mainframe computer system became inadequate. In the 1980s, people proposed "share nothing", a parallel database system, to meet the demand of the increasing data. The share nothing system architecture is based on the use of cluster and every machine has its own processor, storage, and disk. In the late 1990s, the advantages of parallel database was widely recognized in the database field.

However, many challenges on big data arose. With the development of Internet servies, indexes and queried contents were rapidly growing. Therefore, search engine companies had to face the challenges of handling such big data. Google created GFS and MapReduce programing models to cope with the challenges brought about by data management and analysis at the Internet scale. In January 2007, Jim Gray, a pioneer of database software, called such transformation "The Fourth Paradigm". He also thought the only way to cope with such paradigm was to develop a new generation of computing tools to manage, visualize, and analyze massive data.

Over the past few years, nearly all major companies, including EMC, Oracle, IBM, Microsoft, Google, Amazon, and Facebook, etc. have started their big data projects. Big data shows great value in real application and challenges arise. At present, data has become an important production factor that could be comparable to material assets and human capital.

In this paper, we describe the Big Data Management System(BDMS) from a new perspective: the view of a computer. We focus our attention on the system architecture in the BDMS and break it down into several parts to elaborate. Key techniques will be explained to better present the systematic functionality.

[1]http://cloudberry.ics.uci.edu/

The paper is organized as follows. In Section II, we introduce the BDMS from five aspects, input, storage, control, computation and output. Section III introduces the framework and implementation of dater, biggy. The details of manipulation of biggy are given in Section IV. Finally, we conclude with futures in section V.

II. Break Down BDMS

A. Data Input

1) Data Generation: **Web data** is unstructured data from the web, also including social network data, location based service data and the linked pages. Web Data is very complex as compared to traditional text document. **Enterprise Data**, the internal data of enterprises are the main sources of big data. The internal data of enterprises mainly consists of online trading data and online analysis data, most of which are historically static data and are managed by traditional relation DBMSs in a structured manner. **Government Data** are collected from government agencies. Inundated with database schemas, documents, emails, web content and XML, agencies are left with daunting integration challenges. **Other Data** generates from more sources. As scientific applications are increasing, the scale of datasets is gradually expanding, and the development of some disciplines greatly relies on the analysis of masses of data. In addition, pervasive sensing and computing among nature, commercial and social environments are generating heterogeneous data with unprecedented complexity. These datasets have their unique data characteristics in scale, time dimension, and data category.

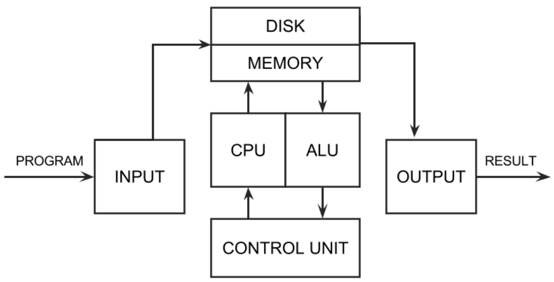
2) Data Feed: Data feed is also known as data acquisition, having continous data arrive into DBMS from external sources and incrementally populate a persisted dataset and associated indexes. In the past, ETL (Extract Transform Load) systems work the same way. It is also similar to stream data processing.

A simple way of having data being put into a Big Data management system on a continuous basis is to have a single program fetch data from an external data source, parse the data, and then invoke an insert statement per record or batch of records. It is hard to reason about the data consistency, scalability and fault-tolerance offered by an assembly 'gluing' together different systems. Therefore, it is natural for a BDMS to provide "native" support for data feed management. In AsterixDB, we build a fault-tolerant data feed facility that scales through partitioned parallelism by using a high-level language. A generic plug-and-play model can help datar cater to a wide variaty of data sources and applications.

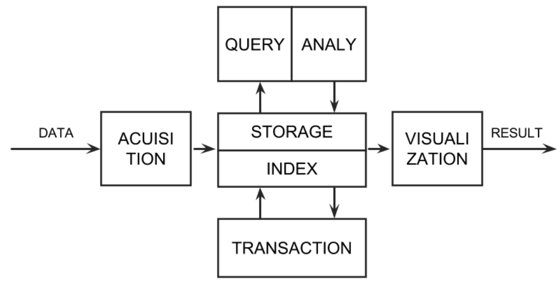
B. Data Storage

1) Data Storage: Big data storage refers to the storage and management of large-scale datasets while achieving reliability and availability of data accessing. Various storage systems emerge to meet the demands of massive data. There are three main NoSQL databases technologies: key-value DB, column-oriented DB and document-oriented DB.

Key-Value DB Key-value Databases are constituted by a simple data model and data is stored corresponding to key-values. Every key is unique and customers may input queried values according to the keys. Such databases feature



(a) Computer Architecture



(b) Datar Architecture

Fig. 2. Computer VS Datar comparison of architecture.

a simple structure and the modern key-value databases are characterized with high expandability and shorter query response time than those of relational databases. Dynamo[2] [14] by Amazon,Voldemort[3] [15] by LinkedIn and Redis[4] [16], Memcache DB[5] [17], Scalaris[6] [18], Riak, Tokyo Canbinet, Tokyo Tyrant.

Column-oriented DB The column-oriented databases store and process data according to columns other than rows. Both columns and rows are segmented in multiple nodes to realize expandability. BigTable[7] [19] by Google, Cassandra[8] [20] by Facebook, Hbase[9] [21] cloned from BigTable, HyperTable[10].

Documnet-oriented DB Compared with key-value storage, document storage can support more complex data forms. Since documents do not follow strict modes, there is no need to conductmode migration. MongoDB[11] [22] open-source, SimpleDB[12] by Amazon, CouchDB[13] [23] by Apache.

Graph-oreinted DB A graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data. Apache Giraph[14], Neo4J[15], OrientDB[16].

In addition, other companies and researchers also have their solutions to meet the different demands for storage of big data. A lot of features include Pluggable Storage Engines (e.g., MySQL) provided by Voldemort. TiDB also supports external database connection. In this situation, storage is just like disk that can be added, replaced and deleted.

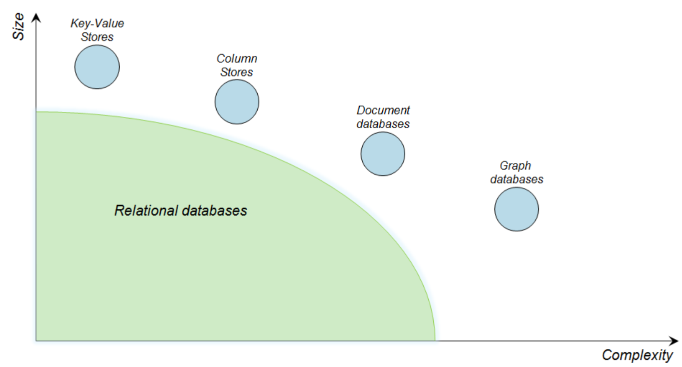


Fig. 3. Comparision of data storage model.

Key-Value	Column	Document	Graph	Other
Dynamo	BigTable	MongoDB	Giraph	
Voldemort	Cassandra	SimpleDB	Neo4j	
Redis	Hbase	CouchDB	OrientDB	
MemcacheDB	HyperTable			
Scalaris				

TABLE I. POPULAR BIG DATA STORAGE SYSTEM.

2) Data Index: Index is always an effective method to reduce the expense of disk reading and writing, and improve insertion, deletion, modification, and query speeds in both traditional relational databases that manage structured data, and other technologies that manage semistructured and unstructured data. However, index has a disadvantage that it has the additional cost for storing index files which should be maintained dynamically when data is updated.

Basic structures include Hash table, Tree-based index, Multidimensional index, and Bitmap index. Big data index has additional requirements, such as parallelisim, easily partitioned into pieces for parallel processing. Artificial Intelligence indexing approaches are so called because of their ability to detect

[2] https://aws.amazon.com/dynamodb/
[3] http://www.project-voldemort.com/
[4] http://redis.io/
[5] http://memcachedb.org/
[6] http://scalaris.zib.de/
[7] https://cloud.google.com/bigtable/
[8] http://cassandra.apache.org/
[9] http://hbase.apache.org/
[10] http://www.hypertable.org/
[11] https://www.mongodb.com/
[12] https://aws.amazon.com/simpledb/
[13] http://couchdb.apache.org/
[14] http://giraph.apache.org/
[15] https://neo4j.com/
[16] http://orientdb.com/

unknown behavior in Big Data. They establish relationships between data items by observing patterns and categorizing items or objects with similar traits. Latent Semantic Indexing and Hidden Markov Model are two popular AI indexing approaches. In Non-AI indexing approach, the formation of indexes does not depend on the meaning of the data item or the relationship between texts. Rather, indexes are formed based on items most queried or searched for in a particular data set.

C. Data Computation

1) Data Query: MapReduce [24], Dryad[17] [25], All-Pairs, Pregel [26], Spark[18] [27] are the popular programming models and excution engines. Many in-memory database are proposed to accellarate the computation.

MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. The Dryad Project is investigating programming models for writing parallel and distributed programs to scale from a small cluster to a large data-center. Pregel is Googles scalable and fault-tolerant platform with an API that is sufficiently flexible to express arbitrary graph algorithms. Apache Spark is a fast and general engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing.

2) Data Analysis: The analysis can be from simple statistic to deep data mining technology. Nowadays, deep learnig has become a trend in analysis of big data. MLlib[19] [28] by Java, Scipy,Theano,Caffe[20] [29] by Python, TensorFlow[21] [30] by C++, Torch.

MLlib is Spark's machine learning library, focusing on learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives. Caffe is a deep learning framework made with expression, speed, and modularity in mind. TensorFlow is an open source software library for machine learning in various kinds of perceptual and language understanding tasks.

D. Data Control

According to CAP theorem, it is not feasible for BDMS to fullfil ACID (Atomic Consistent Isolation Durable) from traditional DB. However, BASE (Basic Availability Soft-state Eventual consistency) is an alternative.CAP theorem says ot is impossible for a protocol to guarantee both consistency and availability in a partition prone distributed system. Most of the NoSQL database system architectures favour one factor over the other.

BigTable, used by Google App engine, and HBase, which runs over Hadoop, claim to be strongly consistent within a data-center and highly available meaning there's an eventual consistency between data-centers. Updates are propagated to all replicas asynchronously. Amazon's Dynamo, Cassandra and Riak instead sacrifice consistency in favor of availability and partition tolerance. They achieve a weaker form of consistency known as eventual consistency updates are propagated to all replicas asynchronously, without guarantees on the order of updates across replicas and when they will be applied.

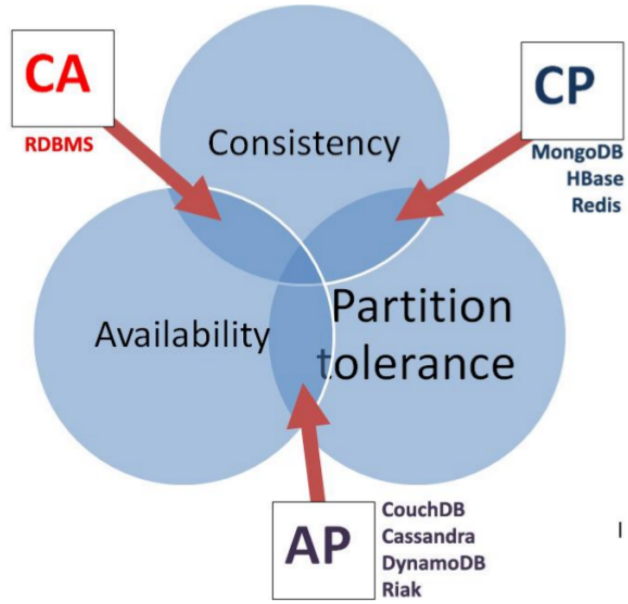


Fig. 4. CAP theorem with associated NoSQL DBs.

1) Data Transaction: In databases, a transaction is a set of separate actions that must all be completely processed, or none processed at all. In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability. It is hard to leverage between the speed and scale of NoSQL, and the relational, transactional strength and consistency of traditional RDBMS.

NewSQL [31] is next-generation scalable relational database management systems (RDBMS) for Online Transaction Processing (OLTP) that provide scalable performance of NoSQL systems for read-write workloads, as well as maintaining the ACID (Atomicity, Consistency, Isolation, Durability) guarantees of a traditional database system. One of the famous NewSQL is Google Spanner [32], which is a globally distributed NewSQL database.

NuoDB[22] that is a distributed database designed with SQL service: all the properties of ACID transactions, standard SQL language support and relational logic. ClustrixDB[23] is a distributed SQL database built for large-scale and fast-growing applications. VoltDB [9] is an insanely fast in-memory database with incredible high read and write speeds. CouchDB read operations use a Multi-Version Concurrency Control (MVCC) model.

Although NewSQL systems vary greatly in their internal architectures, the two distinguishing features common amongst them is that they all support the relational data model and use SQL as their primary interface

[17]https://www.microsoft.com/research/project/dryad/
[18]http://spark.apache.org/
[19]http://spark.apache.org/mllib/
[20]http://caffe.berkeleyvision.org/
[21]https://www.tensorflow.org/
[22]http://www.nuodb.com/
[23]http://www.clustrix.com/

Item	DB	NoSQL	NewSQL
SQL	surpported	non-surpported	surpported
OLTP	not fully supported	supported	fully supported
Trans	ACID	CAP based BASE	ACID

TABLE II. TRANSACTION COMPARISION OF DB, NOSQL AND NEWSQL.

2) Data Recovery: Big data applications as well as operational systems must be supported by a robust and rapid recovery process. As database architecture has fundamentally changed to meet new application requirements, data protection needs to be redefined and re-architected as well.

Big data shook up the database arena, ushering in a new class of "scale out" technologies.The scale-out nature of the architecture can also be difficult for traditional backup applications to handle. Organizations that are deploying big data platforms and applications must realize the importance of backing up their data. Platform-provided mechanisms such as replicas and snapshots are not sufficient to ensure proper data protection and to minimize downtime. Proper backup and recovery requires some investment but is well worth it given the role big data plays in driving business value. Some of the most common mechanisms[24] include:

- Multiple replicas of data eliminates the need for separate backup/recovery tools of big data.
- Lost data can be quickly and easily rebuilt from the original raw data.
- Backing up a petabyte of big data is not economical or practical.
- Remote disaster recovery copies can serve as a backup copy.
- Writing backup/recovery scripts for big data is easy.
- Big Data Backup/Recovery operations costs are very small.
- Snapshots are an effective backup mechanism for big data.

3) Resource Management: Big data computation always runs on thousands of machines, which needs the resource management among clusters. Mesos [33] is a platform for sharing commodity clusters between multiple diverse cluster computing frameworks. The fundamental idea of Hadoop YARN [34] is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons. Apache ZooKeeper [35] is an effort to develop and maintain an open-source server which enables highly reliable distributed coordination.

E. Data Output

1) Data Visualization: Visualization helps us take deep look into the big data, which provides us a interactive and graphic way to embrase the inside of big data. Tools like

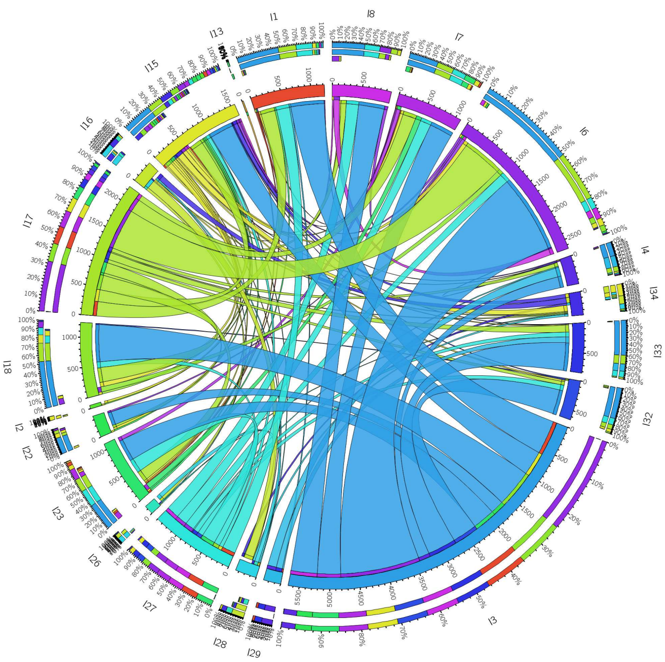


Fig. 5. Big data visualization.

Tableau[25], Plotly[26], Visual.ly[27] are emerging. Zeppelin[28] is a web-based notebook that enables interactive data analytics.

Big Data analytics plays a key role through reducing the data size and complexity in Big Data applications. Visualization is an important approach to helping Big Data get a complete view of data and discover data values. Big Data analytics and visualization should be integrated seamlessly so that they work best in Big Data applications. Many conventional data visualization methods are often used, such as table, histogram, scatter plot, time line, data flow diagram, and entity relationship diagram, etc. Visualizations are not only static; they can be interactive. Interactive visualization can be performed through approaches such as zooming. Scalability and dynamics are two major challenges in visual analytics [36]. Visualization of big data with diversity and heterogeneity (structured, semi-structured, and unstructured) is a big problem.Big data visualization can be performed through a number of approaches such as more than one view per representation display, dynamical changes in number of factors, and filtering.

2) Data Sharing: Because sharing data [37] will necessarily increase the potential benefit to society of the subject's participation by providing greater opportunities for scientific discovery, Brakewood and Poldrack argued that researchers may have an ethical duty to share their data unless doing so would increase risk to the subjects.

In research filed, many insititutions share their data to promote the research. Governments also start share data to public for common benefits. However, some data like personal data and enterprise internal data cannot share since its privacy

[24]http://www.networkworld.com/article/3113036/big-data-business-intelligence/debunking-the-most-common-big-data-backup-and-recovery-myths.html

[25]http://www.tableau.com/

[26]https://plot.ly/

[27]http://visual.ly/

[28]https://zeppelin.apache.org/

System	Input	Storage	Compute	Control	Output
AsterxDB					
HBase					
Spark					

TABLE III. COMPARISION OF POPULAR SYSTEMS COMPONENTS.

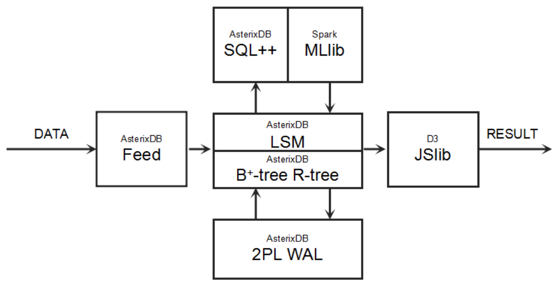


Fig. 6. Framework of biggy.

and confidetial. Therefore, some guidelines our regulations should be made to lead us properly share data, rather than share all or share nothing.

III. BUILD UP DATAR

A. Datar Hypothesis

As we have discussed, we envision a universal architecture of (big) data management, **datar**, to make one more step. The idea essentially comes from computer. A datar is a set of coherent softewares/systems that can manage (big) data pluggablly, automatically and intelligently with specific functionalities, where specific functionalities refer to input, storage, compute, control and output of the (big) data.

B. biggy Framework and Implementation

To put the envisioned datar into practice, we implement it as **biggy**. biggy is based on AsterixDB with several other systems to fulfill the functions of input, storage, compute, control and output. Currently, we plan to implement biggy based on AsterixDB, BAD, Spark-MLlib and d3. AsterixDB is the core component for data storage and control, BAD for data input, Spark-MLlib for data computation and d3 for data output. We make it more pluggable and automatical rather than just gluing them together.The project can be found at Github[29]. Fig. 6 shows the implementation framework of biggy.

Further work of supporting most popular systems (e.g., TensorFlow) as plugins needs done with fullfilment of intelligency.

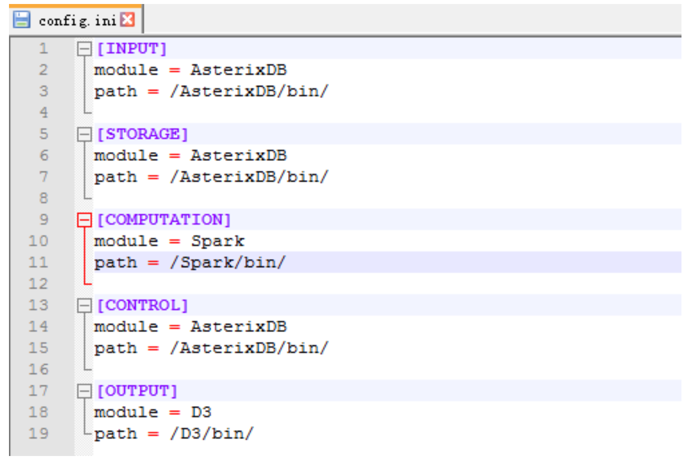


Fig. 7. Configuration of biggy.

Fig. 8. Example of biggy installation and management.

IV. PLAY WITH BIGGY

A. biggy Installation and Management

Fig. 7 shows the configuration of biggy. Fig. 8 shows how the install and manage biggy, in which bigo is an instance of biggy. Management includes *create, start, use, stop, delete and describe*.

B. biggy Manipulation of Instance bigo

Fig. 9 shows the manipulation of bigo, input, store, compute, control and output.

1) bigo Input: Data INPUT is to have continous data arrive into biggy from external sources and incrementally populate a persisted dataset and associated indexes.

2) bigo Store: Data STORAGE is to store the data on physical storage devices.

3) bigo Compute: Data COMPUTATION is to explore valuable information from persisted data by shallow query, data mining and deep learning.

4) bigo Control: Data CONTROL is to make all the data management conducted without conflicts, faults and failures.

5) bigo Output: Data OUTPUT provides a way to show the results.

[29] https://github.com/Ideamaxwu/biggy

- store [create, delete] datastore

```
./bigo.sh use biggy bigo
using biggy bigo
biggy>>> store [-new, -delete] bigdb
store...
*some excuting info here*
stored.
biggy>>> quit
use biggy bigo end.
```

- compute [query, analysis] code_query

```
./bigo.sh use biggy bigo
using biggy bigo
biggy>>> compute [-query, -analysis] use dataverse bigdb; for $ds in dataset Metadata.Dataset return $ds
*biggy>>> compute -query use dataverse bigdb; create type UserType as {userName: string, userAge: int, userGe
*biggy>>> compute -query use dataverse bigdb; create dataset Users(UserType) primary key userName;*
*biggy>>> compute -query use dataverse bigdb; insert into dataset Users({"userName":"Tom","userAge":20,"userGe
*biggy>>> compute -query use dataverse bigdb; for $user in dataset Users return $user;*
*biggy>>> compute -analysis sc.parallelize(1 to 1000).count()*
compute...
*some excuting info here*
computed.
biggy>>> quit
use biggy bigo end.
```

Fig. 9. Example of bigo manipulation.

C. biggy Evaluation

HOW?

V. Conclusion And Future Work

We have illustrated the BDMS from the perspective of a computer in five component and propose biggy as an implementation of datar.

We have entered an era of Big Data. Through better analysis of the large volumes of data that are becoming available, there is the potential for making faster advances in many scientific disciplines and improving the profitability and success of many enterprises. However, many technical challenges described in this paper must be addressed before this potential can be realized fully. The challenges include not just the obvious issues of scale, but also heterogeneity, lack of structure, error-handling, privacy, timeliness, provenance, and visualization, at all stages of the analysis pipeline from data acquisition to result interpretation. Furthermore, these challenges will require transformative solutions, and will not be addressed naturally by the next generation of industrial products. We must support and encourage fundamental research towards addressing these technical challenges if we are to achieve the promised benefits of Big Data.

References

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[2] J. Von Neumann, "First draft of a report on the edvac," *IEEE Annals of the History of Computing*, no. 4, pp. 27–75, 1993.

[3] C. W. Bachman, "On a generalized language for file organization and manipulation," *Commun. ACM*, vol. 9, no. 3, pp. 225–226, 1966.

[4] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[5] M. Stonebraker, E. Wong, P. Kreps, and G. Held, "The design and implementation of INGRES," *ACM Trans. Database Syst.*, vol. 1, no. 3, pp. 189–222, 1976.

[6] M. Stonebraker, "The postgres DBMS," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990.*, 1990, p. 394.

[7] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu, "Mariposa: A wide-area distributed database system," *VLDB J.*, vol. 5, no. 1, pp. 48–63, 1996.

[8] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik, "C-store: A column-oriented DBMS," in *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, 2005, pp. 553–564.

[9] M. Stonebraker and A. Weisberg, "The voltdb main memory DBMS," *IEEE Data Eng. Bull.*, vol. 36, no. 2, pp. 21–27, 2013.

[10] S. Alsubaiee, Y. Altowim, H. Altwaijry, A. Behm, V. Borkar, Y. Bu, M. Carey, I. Cetindil, M. Cheelangi, K. Faraaz *et al.*, "Asterixdb: A scalable, open source bdms," *Proceedings of the VLDB Endowment*, vol. 7, no. 14, pp. 1905–1916, 2014.

[11] R. Grover and M. J. Carey, "Data ingestion in asterixdb." in *EDBT*, 2015, pp. 605–616.

[12] S. Alsubaiee, A. Behm, V. Borkar, Z. Heilbron, Y.-S. Kim, M. J. Carey, M. Dreseler, and C. Li, "Storage management in asterixdb," *Proceedings of the VLDB Endowment*, vol. 7, no. 10, pp. 841–852, 2014.

[13] V. Borkar, M. Carey, R. Grover, N. Onose, and R. Vernica, "Hyracks: A flexible and extensible foundation for data-intensive computing," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 1151–1162.

[14] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 205–220, 2007.

[15] R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, and S. Shah, "Serving large-scale batch computed data with project voldemort," in *Proceedings of the 10th USENIX conference on File and Storage Technologies*. USENIX Association, 2012, pp. 18–18.

[16] J. L. Carlson, *Redis in Action*. Manning Publications Co., 2013.

[17] B. Fitzpatrick, "Distributed caching with memcached," *Linux journal*, vol. 2004, no. 124, p. 5, 2004.

[18] T. Schütt, F. Schintke, and A. Reinefeld, "Scalaris: reliable transactional p2p key/value store," in *Proceedings of the 7th ACM SIGPLAN workshop on ERLANG*. ACM, 2008, pp. 41–48.

[19] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[20] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[21] L. George, *HBase: the definitive guide*. " O'Reilly Media, Inc.", 2011.

[22] K. Banker, *MongoDB in action*. Manning Publications Co., 2011.

[23] J. C. Anderson, J. Lehnardt, and N. Slater, *CouchDB: the definitive guide*. " O'Reilly Media, Inc.", 2010.

[24] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[25] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[26] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: a system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 135–146.

[27] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets." *HotCloud*, vol. 10, pp. 10–10, 2010.

[28] X. Meng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *JMLR*, vol. 17, no. 34, pp. 1–7, 2016.

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[30] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale

machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[31] M. Stonebraker, "Newsql: An alternative to nosql and old sql for new oltp apps," *Communications of the ACM. Retrieved*, pp. 07–06, 2012.

[32] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild *et al.*, "Spanner: Googles globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.

[33] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center." in *NSDI*, vol. 11, no. 2011, 2011, pp. 22–22.

[34] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 2013, p. 5.

[35] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems." in *USENIX annual technical conference*, vol. 8, 2010, p. 9.

[36] L. Wang, G. Wang, and C. A. Alexander, "Big data and visualization: methods, challenges and technology progress," *Digital Technologies*, vol. 1, no. 1, pp. 33–38, 2015.

[37] R. A. Poldrack and K. J. Gorgolewski, "Making big data open: data sharing in neuroimaging," *Nature neuroscience*, vol. 17, no. 11, pp. 1510–1517, 2014.

[38] P. Pirzadeh, M. J. Carey, and T. Westmann, "Bigfun: a performance study of big data management system functionality," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 507–514.

[39] M. J. Carey, S. Jacobs, and V. J. Tsotras, "Breaking bad: a data serving vision for big active data," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*. ACM, 2016, pp. 181–186.

[40] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011, pp. 61–72.