STEM: A Simulation-Based Testbed for Electromagnetic Big Data Management

Mengyuan Lyu[1, 3], Peiquan Jin[1, 2], Zhou Zhang[1, 2], Shouhong Wan[1, 2], Lihua Yue[1, 3]
[1] School of Computer Science and Technology, University of Science and Technology of China
[2] Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences
[3] Science and Technology on Electronic Information Control Laboratory, Chengdu 610036, China
Hefei, China
lmys@mail.ustc.edu.cn, jpq@ustc.edu.cn, zzwolf@mail.ustc.edu.cn, {wansh, llyue}@ustc.edu.cn

Abstract—**With the development of networked radars and wireless communication technologies, electromagnetic data is becoming a new type of big data. Compared with other kinds of big data, such as Internet big data, financial big data, and healthcare big data, electromagnetic big data has some special properties. For example, they usually contain rich and varying labels that describe the features of electromagnetic space. However, existing big-data benchmark tools cannot support the generation and test of electromagnetic big data. In this paper, we aim at providing a simulation-based testbed for electromagnetic big data. The testbed, named STEM, can simulate real-world electromagnetic big data. It supports generating real-time electromagnetic data streams with varying labels. In addition, it is reusable, reconfigurable, and flexible for users to generate workloads for different scenarios. After a brief introduction on the architecture of STEM, we present the implemental details of STEM. Then, we present a case study as well as performance evaluation to demonstrate the usability and flexibility of STEM.**

Keywords- Electromagnetic big data; Testbed; Simulation

I. INTRODUCTION

According to an IDC report, the global data will grow to 163 zettabytes by 2025, which is ten times to the 16.1 zettabytes of data generated in 2016 [1]. The increasing of data volumes in different areas, such as web search [2], social networks [3], moving objects databases [4-5], and electromagnetic spaces, leads to the big data era. Electromagnetic big data is a new type of big data, which is advanced with the development of electromagnetic space networks, networked radars, and wireless communication technologies. Compared with other kinds of big data, such as Internet big data, financial big data and healthcare big data, electromagnetic big data has its own characteristics. First, it contains rich label information, e.g., time and location labels, which is very important for the analysis and use of the data. Besides, unlike the common multi-sensor data whose file size is dozens of kilobytes [6], electromagnetic big data is a kind of single-sourced big data, meaning that each electromagnetic device can produce high-speed data streams with a large volume of data. In an electromagnetic space network, multiple radar devices can simultaneously generate high-speed electromagnetic data flows up to 100Gbps. To address these challenges, a testbed for accurately evaluating the performance of electromagnetic big data related approaches has been a critical and important issue.

Existing benchmarks, such as YCSB [7], BigDataBench [8] and BigBench [9], have their own data generators. However, the generated benchmark data cannot exhibit the characteristics of electromagnetic big data. As a result, it is difficult to accurately measure the performance of electromagnetic big data. Another problem in testing electromagnetic big data is that we lack real environments. So far, it is much hard to construct a real electromagnetic space network that contains multiple satellites and radar devices. Moreover, a specific electromagnetic space network is not able to support the diversity of test requirements.

In this paper, aiming to build a reconfigurable and flexible environment for electromagnetic big data researches, we propose a simulation-based testbed called STEM (*Simulation-based Testbed for Electromagnetic big data Management*). STEM is designed to be reusable and flexible to allow users to customize different electromagnetic environments. In addition, it provides a friendly user interface. The main contributions of the paper are summarized as follows:

(1) We present a simulation-based testbed named STEM for electromagnetic big data. STEM can simulate real-world radar echoes and integrate the characteristics of electromagnetic big data into the simulation process.

(2) The proposed STEM provides a user-friendly interface and flexible configuration options for data generation, which help users to modify field data (such as labels) to satisfy new requirements and customize the electromagnetic environment.

(3) We present a case study of STEM on MongoDB [10] to demonstrate its reusability and configurability.

The remainder of the paper is structured as follows. Section Ⅱ introduces related work. Section Ⅲ presents the architecture of STEM. In Section Ⅳ, we show the key technologies of STEM. Section Ⅴ presents a case study of STEM on MongoDB. And finally we conclude the paper in Section Ⅵ.

II. RELATED WORK

Many existing big data benchmark tools have their own data generators. Some of data generators are extensible, while others are inextensible. For example, LinkBench [11] uses an inextensible data generator which is designed to generate synthetic data with similar characteristics to real social graph data, while BigDataBench uses an extensible data generator

DOI reference number: 10.18293/SEKE2018-083

named BDGS [12], which contains a text generator, a graph generator and a table generator. No matter using which kind of data generator, they both have pros and cons.

Benchmarks with inextensible data generator usually can produce meaningful results for specific application, such as HiBench [13] (for Hadoop system), LinkBench and TPC-DS [14] (for RDBMS). None of them can generate data with similar characteristics to electromagnetic data. For example, HiBench contains eight workloads, which can be classified into four categories: Micro Benchmarks, Web Search, Machine Learning and HDFS Benchmarks. LinkBench, as mentioned before, can generate social graph data like Facebook network. TPC-DS implements a multi-dimensional data generator—MUDD, which is designed for structured data type. For these application-specific limitations, some researchers turn to extensible data generator.

Though extensible data generator can produce various kinds of big data, it still has limitations. First, the data's veracity is questionable. For example, PGDF [15], a parallel data generation framework applied in BigBench, uses distribution functions to generate distributed big data. But its distribution functions' data source is provided by RNG (Random Number Generator). Second, the lack of flexibility is also a common issue. Besides PGDF, another data generator—BDGS, which is implemented on BigDataBench, uses the data model derived from real data sets. But electromagnetic big data contains a wealth of label information and the labels can be changeable for fitting different requirements. A data generator like BDGS, would lead to more redundant work on deriving data models. Another example is YCSB, a popular benchmark for NoSQL system, also have two limitations mentioned above. Its workloads consist of records, each with several fields which are random strings of ASCII characters and it also need to modify the data configuration every time, which is inconvenient.

An ideal way for electromagnetic big data tests is to run tests in a real electromagnetic environment. However, it is costly and hard. Instead, a simulation-based testbed is more suitable for electromagnetic big data researches, due to its flexibility and reconfigurability. Compared with previous methods, our proposal can simulate real-world radar echoes. In addition, it provides a user-friendly interface and flexible configuration options for users to customize the electromagnetic environment. To the best of our knowledge, our proposal is the first testbed that supports the simulation of electromagnetic big data.

III. Architecture of STEM

Fig. 1 shows the architecture of STEM. It mainly consists of three modules: the EES (*Electromagnetic Environment Setting*) module, the SDT (*Simulation and Data Testing*) module, and the TEM (*Tools for Easy Management*) module. The detailed architecture is described as follows.

The EES Module. This module is designed to customize electromagnetic environments. It contains three components: Target Setting, Label Setting, and Device Setting. Every component is designed to be flexible and configurable. Users can add, delete, or edit elements within each component. The Target Setting component provides two operational modes, namely automatic generation and manual setting. This makes the system

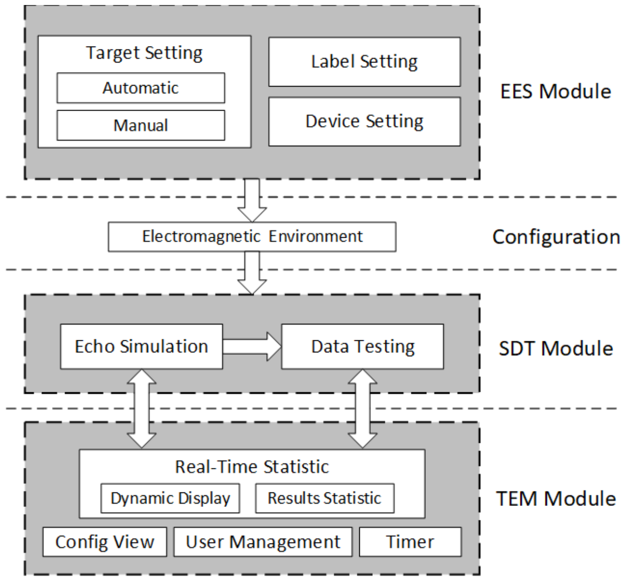


Figure 1. Architecture of STEM

easier for users to operate. After setting up the three components, the *EES Module* can generate a configuration of electromagnetic environment for the *SDT Module*.

The SDT Module. This module includes two sub-modules: the Echo Simulation module and the Data Testing module. The Echo Simulation module plays the role of data generator. It can generate electromagnetic data based on the configuration which is set by the EES Module, and then send them to the Data Testing module. The Data Testing module is designed to evaluate the performance of a database in electromagnetic big data. Currently, we've implemented it on MongoDB, which is the most popular NoSQL database system according to the DB-Engines Ranking [16]. Testing results and the data generated by Echo Simulation module will be sent to the *TEM Module* for visualization.

The TEM Module. This module provides some management tools for easy use. For example, the Real-Time Statistic in the *TEM Module* can display the results produced by the *SDT Module* in line chart and numbers. The line chart can clearly reflect the trend of the results, while the numbers can accurately show the value. Meanwhile, the Real-Time Statistics can convert the electromagnetic data into signal waveform synchronously in order to express it more vividly. In addition, we design some other management tools, such as Config View, User Management and Timer. Config View can help users check the configuration. User Management divides users into administrators and regular user, and grants different authorities. Timer can set the running time of whole system for saving users' time.

IV. Implementation of STEM

In this section, we describe the implemental details of STEM.

A. Implementation of the EES Module

The EES Module is a software layer that simulates the real electromagnetic environments. We divide it into three parts

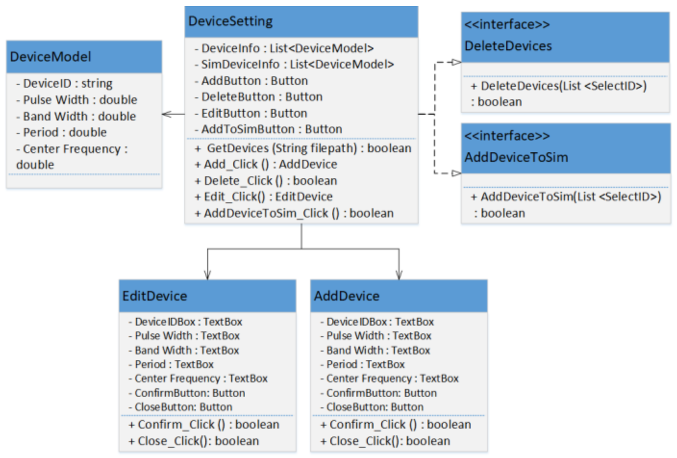


Figure 2. The class diagram of Devices Setting

according to the principle and characteristics of electromagnetic data, which are *Devices Setting, Targets Setting*, and *Labels Setting*. These components provide functionalities with similar interfaces for users to customize the environments. We are going to take the Device Setting as an example to introduce the similar parts of these interfaces. Then, we describe the differences between the Labels Setting and Targets Setting.

Devices Setting. Fig. 2 shows the class diagram of Device Setting. In our models, a radar device is represented as the following structure: {*DeviceID, Pulse Width, Band Width, Period, Center Frequency*}. The DeviceID is used to facilitate access to specific device. Pulse Width, Band Width, Period and Center Frequency are the parameters of a device, which generally won't change. In order to prevent these invariant parameters from being transmitted over and over again, we put device information into the database and use "DeviceID" to replace these parameters during transmission. As for some parameters that can be changed during the work, such as pitching angle and azimuth angle, we treat them as labels and put them in Label Setting module.

In order to make the module flexible and configurable, we offer some interfaces for users to construct they own devices. *AddDevice*() and *EditDevice*(*SelectedID*) provide a new graphical interface for user input, The interfaces and the descriptions are shown in Table 1.

Each interface provides guidance for easy use and can handle users' operation errors, like *DeviceID* is repeated or didn't select the devices to delete. The other two sub-modules offer similar interfaces, like *AddTarget*(), *DeleteLabel*(*List <SelectedID>*) and so on. The differences will be discussed in their own sections.

Targets Setting. Fig. 3 gives a simple class diagram of Target Setting. Considering that there may be a lot of targets needed to be set in one experiment, so we offer two operations mode: Manual and Automatic. The Manual mode is similar to what we use in Devices Setting. It involves the following interfaces like *AddTarget*(), *DeleteTarget*(*List <SelectedID>*) and so on. If there is no requirement on simulating specific targets, we can leave this to the Automatic mode, which can automatically generate the targets.

TABLE I. DEVICE INTERFACES AND DESCRIPTIONS

Interface	Description
AddDevice()	Create a new radar device
DeleteDevice(List <*SelectedID*>)	Delete the radar devices selected by users
EditDevice(*SelectedID*)	Edit a radar device selected by users
AddDeviceToSim(List <*SelectedID*>)	Add the selected devices into the simulation list

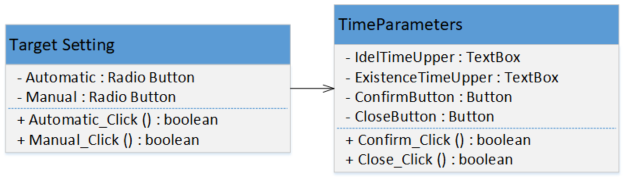

Target Setting
- Automatic : Radio Button
- Manual : Radio Button
+ Automatic_Click () : boolean
+ Manual_Click () : boolean
TimeParameters
- IdelTimeUpper : TextBox
- ExistenceTimeUpper : TextBox
- ConfirmButton : Button
- CloseButton : Button
+ Confirm_Click () : boolean
+ Close_Click () : boolean

Figure 3. The class diagram of Targets Setting

For the sake of simplicity, each target is modeled as a point, which means that we ignore its size and shape. Thus, the model we use to describe a target can be expressed as {*TargetID, Distance, Scattering Coefficient, Velocity*}. In our previous design, we also need to set the appearance and disappearance timing of the target. But it will bring much inconvenience. For example, suppose that an identical target appears intermittently within the radar field of view, we have to repeatedly set its appearance and disappearance timing while other parameters remain the same. Considering this, we remove these two parameters from the target model and offer other two parameters needed to be set only once for users: Idle Time Upper and Existence Time Upper. The Idle Time Upper defines the maximum time between the disappearance of the last target and the occurrence of next target. And the Existence Time Upper defines the maximum time that a target can be exposed within the radar field of view. Therefore, the appearance and disappearance timing of the target are determined by system while users only need to define the boundaries.

Labels Setting. The rich label information is one of the key characteristics of electromagnetic data. The labels can be divided into two categories: some valuable information for the analysis of the electromagnetic data, such as the time and space label, and the parameters that can be changed during the working of a radar device, as mentioned in the Device Setting of this section.

A label can be simply described as: {*LabelID, Description, Bytes, Count*}. Similarly, we design some interfaces like *AddLabel*(), *DeleteLabel*(), *EditLabel*() and *AddLabelToSim*() for user-friendly extension. The difference is that we add a little limitations due to the importance of the time and space labels, for example, the time and space labels cannot be deleted and they are added to the simulation list by default even if not be selected.

Now we can give a general introduction to the final data structure. The electromagnetic data model can be divided into four parts: *Labels*, *IdBytes*, *DeviceID* and *Amplitude*. Fig. 4 gives an example of electromagnetic data.

As mentioned in the Devices Setting of this section, the elements we use to represent a radar device generally remain

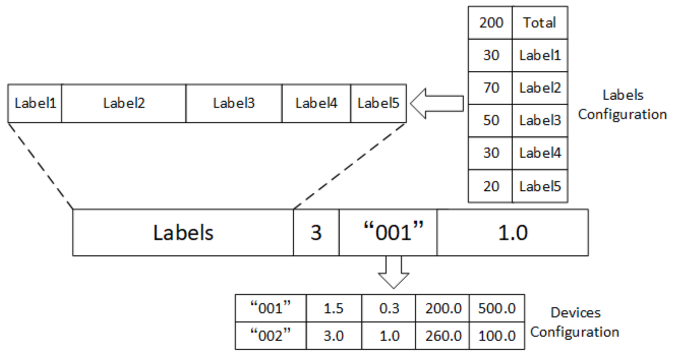


Figure 4. An example of electromagnetic data

stable. Thus, in order to prevent these invariant parameters from being transmitted, we use the *DeviceID* like "001" to replace these parameters for saving space and maintain a *Devices Configuration* in the database for accelerating the search on devices. We also use *IdBytes* to point out the bytes it takes for reading it later, which equals 3 in Fig. 2. *Amplitude* is calculated by the *SDT Module*, we will introduce it in the next section.

The label part is represented by a byte array. After setting up the Labels Setting module, the format of labels will be applied to every piece of data generated by devices. Through this way, we do not need to record the bytes for every label, i.e., the *Labels Configuration* is space efficient, which is similar to the *Devices Configuration*. In the example shown in Fig. 2, we can see from the *Labels Configuration* that the first 200 bytes are the label parts, containing five labels which occupy 30 * 2, 70, 50 and 20 bytes, respectively. The next four bytes are an integer with the value of 3, which means that the next three bytes are the *DeviceID*. The rest is the *Amplitude*.

B. Implementation of the SDT Module

The *SDT Module* is the core module of STEM. The most important component in the SDT module is the echo model. Next, we describe the echo model as well as the echo simulation process.

Echo Model. In the design of radar echo model, we have borrowed some ideas and methods in the [17]. We select the LFM (linear frequency modulation) pulse as the simulation object, which is the most widely used pulse compression signal and not sensitive to the Doppler frequency shift. For the LFM pulse, the radar emit signal can be expressed as in (1), where *rect*() is a rectangle function which defined in (2), T represents pulse width, f_c is center frequency, K is frequency modulation rate and B is band width.

$$S_E(t) = rect(t/T)\exp\left[j2\pi\left(f_c t + Kt^2/2\right)\right],\quad K=B/T \quad (1)$$

$$rect\left(\frac{t}{T}\right) = \begin{cases} 1 & |t/T| \le 1/2 \\ 0 & |t/T| \ge 1/2 \end{cases} \quad (2)$$

Suppose that there is a point target approaching a radar device whose distance to the target is R_0, at a radial velocity v.

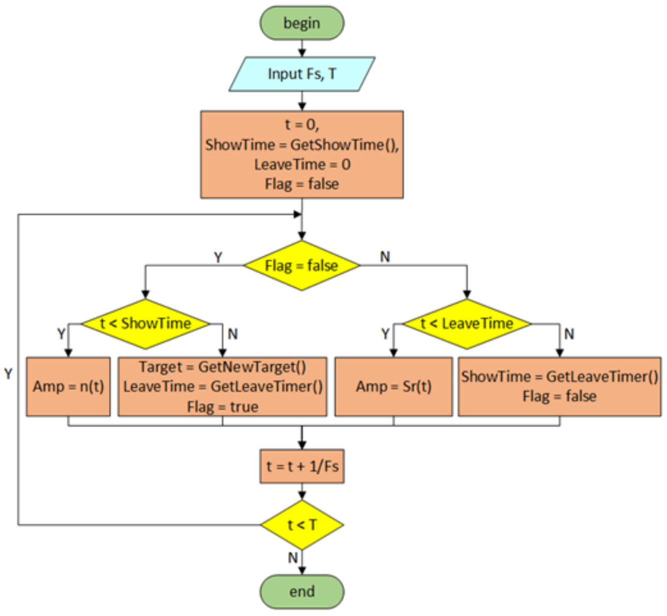


Figure 5. An example of Echo Simulation Process

The pulse repetition frequency of radar is set as T_r. When the nth pulse is emitted, the distance between the target and the radar can be expressed in (3).

$$R^n(t) = R_0 - vnT_r,\quad n = \lfloor t/T_r \rfloor \quad (3)$$

Then, we define the echo by (4). Here, the symbol A represents the scattering coefficient. $\tau(t)$ is time delay function and $n(t)$ is a random function for simulating the environment noise.

$$S_R(t) = AS_E(t - \tau(t)) + n(t),\quad \tau(t) = 2R^n(t)/(c+v) \quad (4)$$

Echo Simulation. Next, we give the process of the echo simulation process. Fig. 5 gives an example with the Timer in the TEM Module and with the choice of Automatic mode in Target Setting of the EES Module. If no Timer is used, the end of system is controlled by the *Pause* Button. If choosing Manual mode in Target Setting, we do not need to get new target during the process.

The parameters we need to input into the system is *Fs* (sampling frequency) and *T* (time limit set by the Timer). *GetShowTime*() and *GetLeaveTime*() are designed to calculate target's appear and disappear time based on the *Idle Time Upper* and the *Existence Time Upper*, which are set in the *Targets Setting*. *Flag* is used to determine if there is a target.

The process in Fig. 5 can be described as follows:

(1) *LeaveTime* and *t* (represents time) is set to zero, *Flag* is set to false and ShowTime is calculate by *GetShowTime*().

(2) If *Flag* = false, which means there is no target, go to (3), otherwise it means that target appears, go to (4).

(3) If $t <$ *ShowTime* (the next target has not appeared yet), the echo that radar has received is just the noise, so Amplitude is determined by $n(t)$. Otherwise it means t is equivalent to or over *ShowTime*, which also means that there would be a new target in the radar range immediately. Thus, we need to design this new target

by using *GetNewTarget*(), calculate its *LeaveTime* and set *Flag* to true. Then, we go to (5).

(4) If $t < LeaveTime$, which means the target is still in the radar range, the echo is calculate by $S_R(t)$. Otherwise it means that this target will disappear. In such a situation, we need to calculate the next target's appear time and set *Flag* to false. Then, we go to (5).

(5) $t = t + 1/F_s$.

(6) If $t < T$, return to (2), otherwise the process is ended.

V. Evaluation

In this section, we describe a case study of using STEM for evaluating the storage performance on MongoDB in electromagnetic big data. The results show that STEM is easy to reconfigure to simulate different electromagnetic environments and evaluate the performance in electromagnetic big data.

A. Experiment Setup

The experiment is based on the network architecture as shown in Fig. 6. It consists of five server nodes and several client nodes. The server nodes constitute a distributed storage cluster for simulating the data receiver and each server node is equipped with Intel 2.1GHz dual CPU, 128GB DDR4 memory, twelve 4T 7200RPM SAS drives and two 240GB SSDs. The client nodes are common computers, mainly used for data acquisition. The server nodes and the client nodes are connected by ten gigabit LAN.

Our STEM runs on the clients of Data Acquisition. First, the client nodes run STEM to set the environment. Then, all the client nodes run the *SDT Module* at the same time to generate data and transmit the data to the server nodes. The states of the servers are monitored by STEM.

B. Use Cases of STEM

We present two cases of using STEM. In the first case, the electromagnetic environment set by each client node is ten radars, automatic targets and 300 labels (each is 50 KB, the file size limit in MongoDB is 16 MB), we test the throughout rate of MongoDB under different number of processes.

The second one is designed to test the throughput of MongoDB under different numbers of labels. Its environment is set to ten radars, automatic targets and ten processes. For the sake of simplicity, all the labels are set to having the same size (50 KB). Both experiments are run for five minutes.

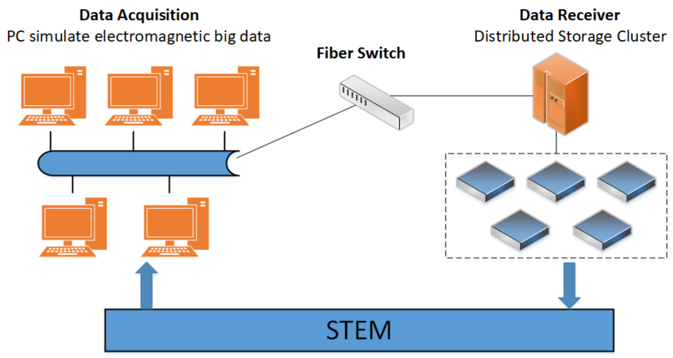
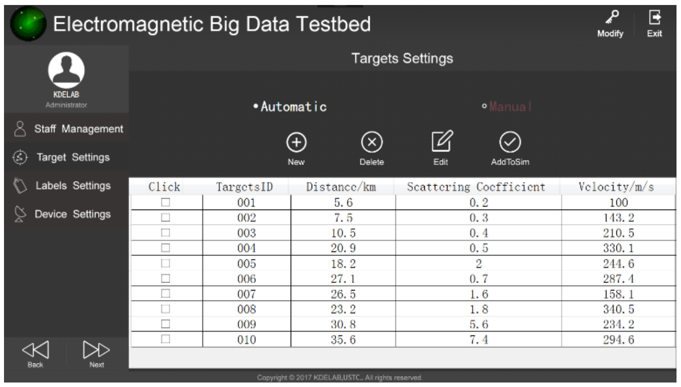

Data Acquisition
PC simulate electromagnetic big data
Fiber Switch
Data Receiver
Distributed Storage Cluster
STEM

Figure 6. Experiment Network Deplonent Architecture

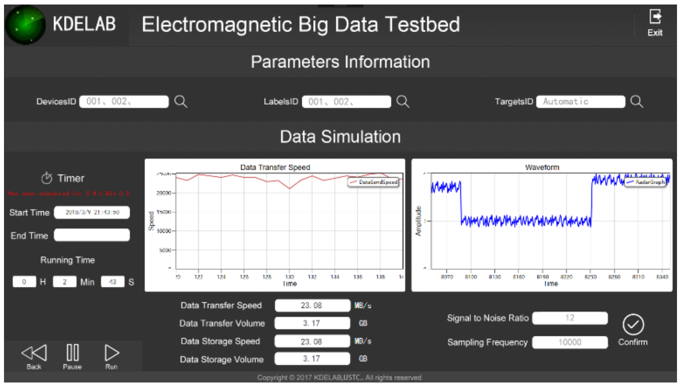


•Automatic ∘Manual

Click	TargetsID	Distance/km	Scattering Coefficient	Velocity/m/s
☐	001	5.6	0.2	100
☐	002	7.5	0.3	143.2
☐	003	10.5	0.4	210.5
☐	004	20.9	0.5	330.1
☐	005	18.2	2	244.6
☐	006	27.1	0.7	287.4
☐	007	26.5	1.6	158.1
☐	008	23.2	1.8	340.5
☐	009	30.8	5.6	234.2
☐	010	35.6	7.4	294.6

Copyright © 2017 KDELAB,USTC.. All rights reserved.

Figure 7. The setting interface of STEM



DevicesID 001、002、
LabelsID 001、002、
TargetsID Automatic

Start Time 2018/3/9 21:43:50
End Time
Running Time 0 H 2 Min 43 S

Data Transfer Speed 23.08 MB/s
Data Transfer Volume 3.17 GB
Data Storage Speed 23.08 MB/s
Data Storage Volume 3.17 GB

Signal to Noise Ratio 12
Sampling Frequency 10000

Copyright © 2017 KDELAB,USTC.. All rights reserved.

Figure 8. The running screenshot of STEM

The experimental process consists of three steps:

(1) Every client node finish the experiment environment setting, click the Next button to the running page;

(2) Every client node sets the Timer to five minutes, then click the Run button to run the system;

(3) Just wait for five minutes and click the Back button to the setting page, reset the environment setting, and back to (2).

Fig. 7 shows the setting page of STEM, and Fig. 8 shows the running screenshot of STEM.

C. Results

Fig. 9 shows the highest throughput (top throughput) as well as the average throughput under different processes. We can see that when the number of processes reaches 13, the top throughput reaches 1126 MB/s, which is close to the upper-bound of the network speed. Basically, the top throughput increases linearly with the increase of the process and the average throughput reaches the bottleneck when the number of processes reaches 11, which is nearly 788 MB/s.

Fig. 10 shows the throughput under different numbers of labels when the number of processes is 10. The average throughput increases with the increasing of the numbers of labels. On the other hand, the top throughput does not show an apparent trend. Specially, the top throughput reaches 1105 MB/s when the number is 20, which is close to the result of 11 processes and 300 labels. A possible reason is that the small numbers of labels make the number of files that can be generated and inserted per second increase, as shown in Fig. 11.

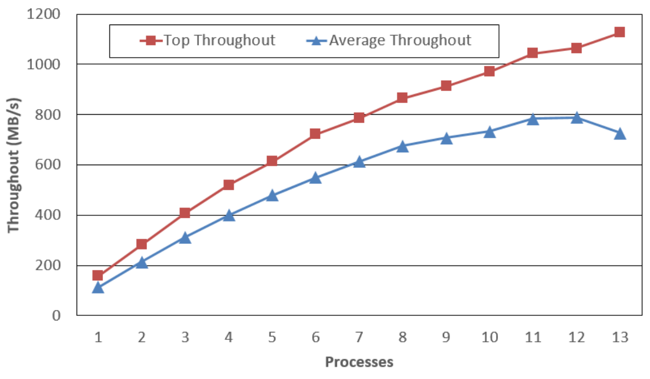


Figure 9. Throughout under different processes

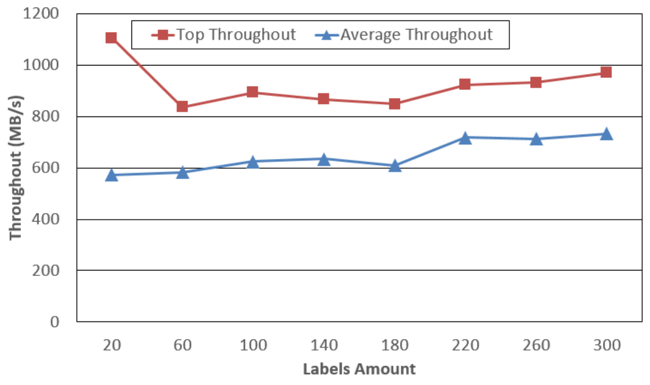


Figure 10. Thoughout under different size of labels

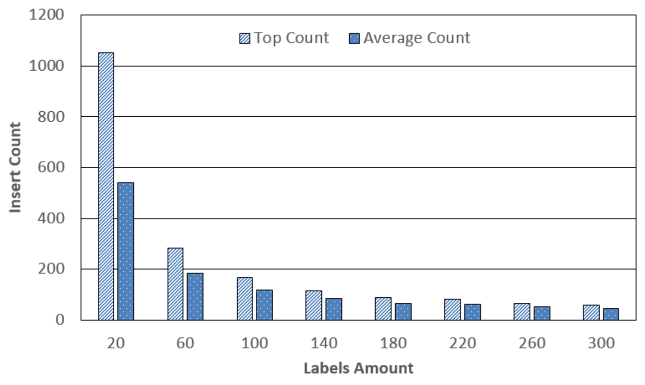


Figure 11. Insert Count under different size of labels

VI. Conclusions and Futrue Work

In this paper, we propose a simulation-based testbed for electromagnetic big data researches, which aims to provide effective support for evaluating the performance of electromagnetic big data management, such as throughput test and real-time processing measurement. The proposed testbed is designed to be reusable, flexible, and reconfigurable. Currently, STEM is able to evaluate the storage performance of electromagnetic big data on MongoDB, such as the test of write throughput, the average throughout, and the insert counts per second.

Our future work will concentrate on supporting multiple types of databases, offering more kinds of performance tests, and improving the efficiency of data generation. We plan to support HBase, Cassandra and other popular database systems in future. In addition, we will adopt other performance tests like query processing to make the testbed multifunctional and the parallel computing on every radar to make data generation more efficient.

Acnowledgement

This work is supported by the National Science Foundation of China (61672479 and 61472376), and a fund from the Science and Technology on Electronic Information Control Laboratory. Peiquan Jin is the corresponding author.

References

[1] D. Reinsel, J. Gantz, and J. Rydning, "Data Age 2025: The evolution of data to Life-Critical," https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf, Accessed in April 2018.

[2] J. Zhao, P. Jin, Q. Zhang, and R. Wen. "Exploiting location information for Web search". Computers in Human Behavior, 2014, 30: 378-388.

[3] L. Zheng, P. Jin, J. Zhao, and L. Yue. "A fine-grained approach for extracting events on microblogs", International Conference on Database and Expert Systems Applications (DEXA), 2014, pp. 275-283

[4] P. Jin, L. Zhang, J. Zhao, L. Zhao, and L. Yue, "Semantics and modeling of indoor moving objects", International Journal of Multimedia and Ubiquitous Engineering, 2012, 7 (2): 153-158

[5] C. Huang, P. Jin, H. Wang, N. Wang, S. Wan, and L. Yue. "IndoorSTG: A flexible tool to generate trajectory data for indoor moving objects., 2013 IEEE 14th International Conference on Mobile Data Management (MDM), 2013, pp. 341-343

[6] X. Hao, P. Jin, and L. Hua, "Efficient Storage of Multi-Sensor Object-Tracking Data," IEEE Transactions on Parallel and Distributed Systems, 2015, 27(10): 2881-2894

[7] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," ACM Symposium on Cloud Computing (SoCC), 2010, pp. 143-154.

[8] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, et al, "Bigdatabench: a big data benchmark suite from internet services," IEEE International Symposium on High Performance Computer Architecture (HPCA), 2014, pp. 488-499.

[9] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, et al, "BigBench: towards an industry standard benchmark for big data analytics," ACM SIGMOD International Conference on Management of Data (SIGMOD), 2013, pp.1197-1208.

[10] MongoDB. https://www.mongodb.com/.

[11] T. G. Armstrong, V. Ponnekanti, D. Borthakur, M. Callaghan, "LinkBench: a database benchmark based on the Facebook social graph," ACM SIGMOD International Conference on Management of Data (SIGMOD), 2013, pp.1185-1196.

[12] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, et al, "BDGS: a scalable big data generator suite in big data benchmarking," Workshop on Big Data Benchmarks, 2013, pp. 138-154.

[13] S. Huang, J. Huang, J. Dai, T. Xie and B. Huang, "The HiBench benchmark suite: characterization of the MapReduce-Based data analysis," IEEE International Conference on Data Engineering (ICDE) Workshops, 2010, pp. 41-51.

[14] R. O. Nambiar and M. Poess, "The making of TPC-DS," International Conference on Very Large Databases (VLDB), 2006, pp. 1049-1058.

[15] T. Rabl, M.Frank, H. M. Sergieh and H. Kosch, "A data generator for cloud-scale benchmarking," Technology Conference on Performance Evaluation and Benchmarking, 2010, pp. 41-56.

[16] DB-Engines Ranking. https://db-engines.com/en/ranking.

[17] M. A. Richards, "Fundamentals of Radar Signal Processing, Second Edition", 2017.