Enhanced Distributed Data Mining Application using the PostgreSQL Database Management System

Pupezescu Valentin[1], Dragomir Marilena-Cătălina[2]

(1) Electronics, Telecommunications and Information Technology Faculty, Polytechnic University of Bucharest, Bd. Iuliu Maniu, Bucharest, ROMANIA, E-mail: vpupezescu[at]yahoo.com
(2) Electronics, Telecommunications and Information Technology Faculty, Polytechnic University of Bucharest, Bd. Iuliu Maniu, Bucharest, ROMANIA
E-mail: catalina.dragomir[at]protonmail.com

Abstract

Due to the current COVID pandemic, virtual laboratories are becoming increasingly important in the educational process. To support the online learning we developed the third version of a distributed machine learning application that allows the participating students to extract useful information from the PostgreSQL servers alongside other major database management systems that were used in the previous versions(Microsoft SQL Server, MySql and MongoDB). The data sets are distributed among multiple distributed PostgreSQL servers arranged in a master-slave topology. In this version, the data was imported from the MySql database management system. In our experiments from the class we use three data sets: iris1, wine1, conc1. The entire system will function like a Distributed Committee Machine that mines simultaneously through distributed data stored on servers in order to achieve better classifying results compared with the centralized case in which we have only one neural network and one data set. The application provides three neural networks: the classical multilayer perceptron, the pulsating multilayer perceptron and the autoresetting multilayer perceptron. Besides manipulating neural networks, in this virtual lab, the students will deal with some preprocessing operations on data (importing and data arrangement) from the KDD process and they will also learn how to configure the replication of the data sets into the chosen distributed topology.

Keywords: Elearning, Distributed Data Mining, Distributed Relational Database Management Systems, Machine Learning, Replication

1 Introduction

As a consequence of the COVID-19 pandemic, most countries around the world temporarily closed the educational institutions with the hope of stopping the new virus.

An unprecedented number of schools, universities and students were affected worldwide by the current situation. This research was done in order to be applied in the databases for scientific applications laboratory in our university. It can be used in classes of about twenty students. If it is needed, the entire experiment can be simulated individualy by students with virtual machines.

This paper presents the further improvments that were brought to the previous eLearning application presented in our research(Pupezescu, V., Dragomir, M., 2019). In this new version we added the support for the PostgreSQL database management system. The functionality of the Distributed Committee Machine (DCM) was kept and the entire system is functioning using the replication capabilities offered by PostgreSQL. The neural structures that were used in experiments were the autoresetting multilayer perceptrons (Pupezescu, V., 2017).

2 Application Architecture

The Figure 1 represents the architecture of the DCM application while in Figure 2 we represented the theoretical model of DCMs. On the PostgreSQL master server we have the combiner module that is functioning based on the “winner takes all” policy. On every distributed node we have Java TCP or UDP servers (in our experiments we used TCP servers) that will wait for the initial parameters of neural networks. After setting up the configuration of the neural structures, the server will send the data to the distributed machines. Every system will compute and write the missclasification rate in the PostgreSQL master node. Through the replication process the PostgreSQL master server will replicate the written data to the slave nodes. By doing this, the system makes sure that everyone from the class will have access to the final classification results. Furthermore, each server will also write the results locally in Excel files for further processing. An important observation is that the entire databases from the slave nodes are read-only replicas. At this moment we did not implemented multimaster replication.

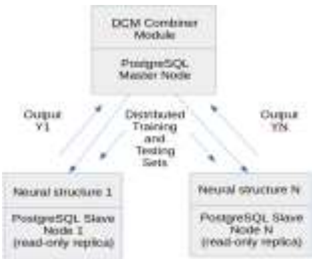

Figure 1. The implementation of DCM architecture with the PostgreSQL DBMS

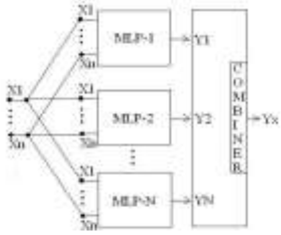

Figure 2. Distributed Committee Machine (Pupezescu, V., Rădescu, R., 2016)

The entire application was developed in the Ubuntu Linux 20.04 Operating System in the Java programming language. The implementation was done in Eclipse IDE 2020-06 (4.16.0). All the distributed experiment for this paper was realized using two Linux machines (Ubuntu 20.04) that have PostgreSQL DBMS installed. The host machine will act as the DCM Combiner module.

3 Data Preprocessing for the PostgreSQL DBMS

The students that will attend to this virtual lab will have the possibility to choose from a variety of data sets. The most important sets that can be mined will be Iris1, Wine1 and Conc1 data sets (https://archive.ics.uci.edu/ml/datasets/iris, https://archive.ics.uci.edu/ml/datasets/wine). The Conc1 set is a self generated data set – in this set all the data is arranged in a concentric manner.

Figure 3. Available data sets for the classification task in the application

The preparation of the data sets involved their import from the MySQL database management system(DBMS) to the PostgreSQL DBMS.

The importing process is quite time consuming because it involves the creation of dedicated tables for each data set. All the data is exported from MySQL in CSV format at the first step.

Afterwards, all the data is imported from individual CSV files to the new created PostgreSQL tables with the import tool available in PostgreSQL 12.

When importing, the students must select the option “Header” because the initial CSV file contains the columns of the table on the first row(this row will be ignored at the importing process).

Below(in Table 1), we present the metadata for the main data sets that were used in the application:

Table 1. iris1, wine1 and conc1 data sets (Pupezescu, V., 2016)

iris1	trr	tsr	trs	tss
Lines	100	50	100	50
Columns	3	3	4	4
wine1	trr	tsr	trs	tss
Lines	90	88	90	88
Columns	3	3	13	13
conc1	trr	tsr	trs	tss
Lines	200	100	200	100
Columns	1	1	2	2

Because PostgreSQL is a relational DBMS, all the data will be stored in classic table format. Besides the tables that contain the actual data we have also two more tables that store the classification results (“rezultate”)and the metadata for each classification problem(“configuratie”).

Once the students finished to import the data sets they can continue by implementing the replication process on all the PostgreSQL servers. In this example we will show how this operation is achieved (Schönig, H. 2015; Shaun, T. 2020) using three PostgreSQL servers: one is installed on the host and the other two(named Ubuntu1 and Ubuntu2) will run inside Oracle VM VirtualBox. Before starting to configure the PostgreSQL servers the students first must verify that all the operating systems are able to communicate with each other. The IPs are assigned as follows:

Table 2. IP allocation among the distributed systems

Sy tem	The master system	The first slave system (Ubuntu1)	The second slave system (Ubuntu2)
IP	192.168.1.7	192.168.1.4	192.168.1.5

The configuration for the replication process is made in teams by the students in two major steps: in the first one team will configure the master system; in the second step, every other student will configure its allocated system(in our case we used just two PostgreSQL slave servers in Oracle VirtualBox for experimental purposes):

Master configuration steps:

1. On the PostgreSQL master we must create a dedicated user for the replication process(note that the password should be a strong passphrase):

sudo su postgres – this command will allow the user to acces the postgres user.

psql – with this command we enter in terminal on their postgresql servers.

CREATE USER replication REPLICATION LOGIN CONNECTION LIMIT 1 ENCRYPTED PASSWORD 'REPLICATION_PASSWORD';

2. In order to increase the number of accepted connection we will use the following command:

ALTER ROLE replication CONNECTION LIMIT -1;

At this moment, if we check(with \du command in a psql connection – see Figure 4) the user list in PostgreSQL we should see that we do not have a connection limit:

List of roles

Role name	Attributes	Member of
pgloader_pg	Superuser, Create role, Create DB	{}
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	{}
replication	Replication	{}

Figure 4. Output for the "\du" command

3. In this step, the students must edit the main configuration file from the PostgreSQL server – "postgresql.conf"; they will open a new terminal and type the following command:

sudo gedit /etc/postgresql/12/main/postgresql.conf

In this file they must find, uncomment and modify four lines as follows:

listen_addresses = 'localhost,192.168.1.7'
wal_level = replica
max_wal_senders = 10
wal_keep_segments = 64

3. Another configuration file that must be modified is "pg_hba.conf":

sudo gedit /etc/postgresql/12/main/pg_hba.conf

At the end of the file, the students must add the ip addresses of the slave servers:

host replication replication 192.168.1.5/0 md5
host replication replication 192.168.1.4/0 md5

Above those lines, the students must also add these lines(# IPv4 local connections:):

host all all 192.168.1.7/0 md5
host all all 192.168.1.5/0 md5
host all all 192.168.1.4/0 md5

After this step, the PostgreSQL master server must be restarted with the following command:

sudo service postgresql restart

The command for checking the status of the server is the next one(the status should be active) :

sudo service postgresql status

Note that is this commands are given in a rapid succesion in the same terminal window it will not be necessary to write the key word "sudo" at each command.

Slaves configuration steps:

1. Before starting the configuration of slaves the PostgreSQL servers must be stopped. On each slave server the students must run this command:

sudo service postgresql stop

With the following command they will also check the status of the server:

sudo service postgresql status

2. Now the student teams can start modifying the main postgresql configuration files. Every student that operates on PostgreSQL slave servers will run this command:

sudo gedit /etc/postgresql/12/main/postgresql.conf

The following lines must be uncommented and modified with the slave IP address(in our case we changed the IP address for the other slave postgresql server – 192.168.1.4):

listen_addresses = 'localhost,192.168.1.5'
wal_level = replica
max_wal_senders = 10
wal_keep_segments = 64
hot_standby = on

3. For every slave server we also must edit the "pg_hba" configuration file:

sudo gedit /etc/postgresql/12/main/pg_hba.conf

With the following line, the students will grant access to the replication user from the master postgresql server:

host replication replication 192.168.1.7/0 md5

Above these lines there is also a section(# IPv4 local connections:) where we must add these lines (for the other server we put 192.168.1.4/0 at first line) and save the file:

host all all 192.168.1.5/0 md5

host all all 192.168.1.7/32 md5

4. In the next step we will enter with the root privileges in the /var/lib/postgresql/12/main/ directory:

sudo su

cd /var/lib/postgresql/12/main/

5. The students that configure the slave systems must delete all content from that directory: sudo rm -rfv *

6. All the content from the master postgresql server will be copied on each slave server (in a new terminal they will enter with root privileges on the postgres user):

sudo su postgres

pg_basebackup -h 192.168.1.7 -U replication -p 5432 -D /var/lib/postgresql/12/main/ -Fp -Xs -P -R

7. In another terminal window the students will start and check the status of their postgresql servers:

sudo service postgresql start

sudo service postgresql status

In order to be able to start the eLearning application all the students must edit the host file for every distributed system. Every machine from the distributed system(in our case the servers are named ubuntu 1 and ubuntu 2) must be present in this file:

sudo gedit /etc/hosts

On the master system we will add the following lines:

192.168.1.4 ubuntu2

192.168.1.5 ubuntu1

On the slave systems we add the master server(named P775): 192.168.1.7 P775

After all the configuration process all the students must check that the PostgreSQL servers are synchronized. In Figure 5 we checked the result table on all servers and verified that the replication process is functioning (these results are for the experiments that were done in the next section – we displayed only five rows from a total of 2000 rows):

	epoca [PK] bigint	pcicg double precision	durata bigint	ip [PK] character varying	durataantrenare bigint
1	0	46	990	192.168.1.4	31
2	0	60	82	192.168.1.5	0
3	1	43.99999999999999	1002	192.168.1.4	43
4	1	38	86	192.168.1.5	4
5	2	14.000000000000002	1008	192.168.1.4	49

Figure 5. The query results on all servers are the same for the "result" table

4 Experimental Distributed Committee Machine v1.2

Like in our previous work (Pupezescu, V., Dragomir, M., 2019), we added a new section in the application dedicated to the PostgreSQL execution(in the index.jsp page). In this new section we are able to set and modify the same parameters like in our previous version (see Figure 6). In the "start_postgres.jsp" page we also added for the students the architecture image of the entire distributed application.

In Figure 6 and Figure 7 we show the interface that allow students to set the initial neural network parameters, the IP addresses and ports of the slave PostgreSQL servers that will receive the replicated data.

As we can see in Figure 8, students will be notified if the distributed experiment starts. After all the participating student teams in the experiment will finish the experiment, the team from the master server will press the "Continue" button in order to display the best classification results obtained inside the DCM(see Figure 9).

Furthermore, the application will query for the final results for each system in the database after the IP address and automatically generate a chart with the missclassification rates obtained on all systems(see Figure 10 – distributed run for 1000 training and testing epochs for the iris data set, 4 neurons on hidden layer, pulse=100).

Figure 6. The start_postgresql.jsp page

Figure 7. Setting the IP addresses and ports for the slave PostgreSQL servers

As we mentioned the implemented DCM application has a special module that automatically writes locally in excel files (on every slave node) all the classification results. This is very useful because they can take and independently analyze the obtained results. They can even import those data in other frameworks if they wish.

Figure 8. Acknowledgment for sending the startup configuration parameters

Figure 9. Distributed results for the classification task

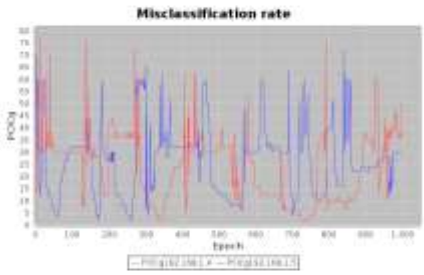

Figure 10. The final classification results obtained on all distributed systems

5 Conclusions

This paper presents an improved Data Mining eLearning application with the new added support for the PostgreSQL database management system. This version makes use of the fast replication process offered by PostgreSQL. All the data sets that were analyzed were replicated among all the distributed PostgreSQL slave nodes. Every node worked to achieve the classification task and wrote the classification results at each testing epoch onto the PostgreSQL master node. Through the replication process all distributed systems had access to the same data thus ensuring the redundancy and high availability for analyzed data sets.

Also, the application offered a dedicated module that exported in excel format on each system the local classification results as well as the automatic display on the master system of the classification results obtained on all distributed systems from the DCM architecture.

In the next version we will implement the multimaster replication offered by PostgreSQL for further optimizations of the DCM architecture. The purpose of this paper is to help students and teachers to cope with the current situation and to continue the educational process and research in areas such as Elearning, Distributed Data Mining, Knowledge Discovery in Distributed Databases and Machine Learning.

References

Pupezescu, V., Dragomir, M. (2019), *Enhanced Elearning Application for Data Mining in a NoSQL Distributed Database Management System* , 14th International Conference on Virtual Learning (ICVL-2019), pp.476-482, Romania, 2019.

Pupezescu, V., (2017), *Auto Resetting Multilayer Perceptron in an Adaptive Elearning Architecture*, Proceedings of The 12th International Conference on Virtual Learning (ICVL-2017), pp.311-317, Octomber 28, Sibiu, ISSN: 1844-8933, 2017.

Pupezescu, V., Rădescu, R. (2016), *The Influence of Data Replication in the Knowledge Discovery in Distributed Databases Process*, ECAI 2016 – International Conference – 8th Edition, 30 June – 02 July, Ploieşti, ROMÂNIA, 2016.

Pupezescu, V., (2016), *Distributed neural structures in adaptive eLearning systems*, Proceedings of the 11th International Conference on Virtual Learning(ICVL-2016), 2016.

https://archive.ics.uci.edu/ml/datasets/iris, accessed 2020.

https://archive.ics.uci.edu/ml/datasets/wine, accessed 2020.

Shaun, T. (2020), *PostgreSQL 12 High Availability Cookbook Third Edition*, Packt Publishing Ltd., ISBN 978-1-83898-485-4, Birmingham.

Schönig, H. (2015), *PostgreSQL Replication Second Edition*, Packt Publishing Ltd., ISBN 978-1-78355-060-9, Birmingham.