ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 4, No. 2, 2019

METHOD FOR TIME MINIMIZATION OF API REQUESTS SERVICE FROM CYBER-PHYSICAL SYSTEM TO CLOUD DATABASE MANAGEMENT SYSTEM

Nataliya Pavych, Tetyana Pavych

Lviv Polytechnic National University, 12, S. Bandery Str., Lviv, 79013, Ukraine
Author's e-mail*: nataliia.y.pavych@lpnu.ua*

Submitted on 01.10.2019

Abstract: Authors have analyzed the current state of cloud calculations and discovered the necessity of Application Programming Interface (API) usage in some of the cyber-physical systems (CPhS). The expediency of creating tools to reduce service time of Application Programming Interface requests and effective synchronization of the local and cloud database has been established. The authors clarified one of the main features of data replication. The expediency of using counter generations in the replication process instead of using system timer has been justified. The authors proposed the asynchronous method of accelerated service time for API requests to cloud database management systems using a synchronization accumulative table and registering changes in database with two-step sets of generations. Library that provides implementation of asynchronous API requests for Salesforce cloud management systems has been developed. Any Ruby on Rails application can use this library. Authors have evaluated and carried out benefits for the proposed solutions in test cases. Results of the test cases confirm service time minimization of API requests to the cloud database management system based on the proposed asynchronous method.

***Index Terms*: API requests, cloud database, cyber-physical systems, database management systems, minimization method, service time.**

I. INTRODUCTION

Nowadays there is a huge development and implementation of CPhS (cyber-physical systems). CPhS is a combination between physical processes and cyber components that provide an organizing of calculation-measuring processes, protected storage, and exchange of business information, influencing physical processes [1]. Internet is one of the most important components of such cyber systems. The analysis of principles of cooperation between variety of components may help in finding out usage expediency of cloud computing. Cloud computing is a model that provides comfortable net access in demand of certain configuration and calculation stock (for example, net data exchange, servers, data storage devices, apps and servers altogether and in-separate) which can be promptly provided with the minimum operating data or provider request [2]. Cloud computing is an effective way for distributed calculations. In addition, cloud computing effectively increases the functional abilities of computer systems and networks. High usage of cloud computing leads it to become the base for a variety of innovations and calculations of complex tasks. What is more, cloud computing became popular in different computer companies and ordinary users. Researchers agree on the idea that cloud computing is one of the most perspective ways in the development of computer calculation at all [2], [3]. Therefore, research about cloud computing applications is highly relevant.

II. ANALYSIS OF RECENT PUBLICATIONS

Cloud computing includes CPhS usage of various hardware and software net servers, methods and instruments, as an Internet service is given to a user for project performance or in order to solve different applied tasks. The main component of such technologies is cloud databases (CDB).

Cloud computing provides easy environment for data storage and information processing. In addition, cloud computing combines hardware tools, licensed program software, communication channels and technical support [3].

Cloud computing is a certain base vector that is given because of the synthesis of various technologies and approaches [3]. We can state that the main components of cloud computing are infrastructure, platform, and program software. Infrastructure is a set of certain physical devices (servers, information exchange channels and external storage devices). The platform is a set of services and software available upon user requests [4].

Some of the main functions of cloud technologies are the following [3], [4]:

- Access to personal information from any computer connected to the Internet.
- Ability to work with information from any device (PC, laptop, tablet, smartphones).
- Independence from any of the operating systems as web-services work in any browser on the PC.
- Information can be viewed and edited from different devices at the same time.
- Various paid programs are for free (or cheaper) web – applications.
- Prevent information loss, as it is stored on cloud storage.
- "Fresh" and updated information.
- Usage of the last updates and applications.

- Ability to unite information with other users.
- Easy to share information with users from any part of the world.

Some of the disadvantages might be [3], [4]:

- Necessity in web-connection. Users should be connected to the Internet in order to get access to the cloud services.
- Software limitation. There are some limitations which can be developed on the cloud and be given to users. User has limitations within the software and does not have an opportunity to set it up for his / her own goals.
- Confidentiality. Nowadays confidentiality of data stored on public clouds causes a lot of controversies, however, in most cases experts state that it is not recommended to store the company's most valuable data on public cloud, as there is no technology that would guarantee 100% confidentiality.
- Security. Cloud is reliable enough, however, in case of attack, hacker gets access to huge data storage. Another disadvantage is the usage of virtualization systems that use the core of standard operating systems (for example, Windows) that allows computer viruses to penetrate the system and lead to vulnerability of the whole system.
- Costly equipment. It is necessary to allocate considerable material resources in order to build own cloud which is not profitable for new small companies.
- Further monetization of the resource. There is a big possibility that a company will increase costs for the services provided.

Cloud database is a database which runs on cloud calculation platforms such as Amazon EC2, GoGrid and Rackspace.

There are two most common deployment models: users can buy directly database access services served by a cloud provider or they can manage databases independently in the cloud with a concept of a virtual machine. There are SQL-oriented and NoSQL models among cloud database models.

Several approaches are used to load and manage database on a cloud.

Concept of a virtual machine – cloud platforms allow users to have an instance of a virtual machine for a limited time and users can manage database in this virtual machine. Users can download a concept themselves from the installed database or use concepts previously created and those which already include optimized installation. For example, Oracle provides machine concept with Oracle Database Enterprise Edition installation on Amazon EC2 and Microsoft Azure.

Database as a Service (DBaaS) – certain cloud platforms provide operations for using a database as a service without physical launch of a virtual machine for the database. Therefore, owners of the application do not need to install and "support" database. However, a service provider takes responsibility for database installation and support. In addition, owners of the application pay according to application usage. For example, Amazon Web Services provide three types of database services as a part of an own cloud deal: Amazon SimpleDB, NoSQL, key-store storage: Amazon Relational Database Service, database service for SQL with MySQL interface and Amazon DynamoDB. In this way, Microsoft provides Azure SQL Database service as a part of its cloud deal [5].

Another option includes database managed hosting where cloud database provider doesn't offer DBaaS, but hosts database and manages it from the name of an owner [5].

Database services consist of components that manage main database instances using an application programming interface (API) of current service. API services are available for end-users and allow them to perform operations of size changes on their own database copies. For example, Amazon Relational Database Services APIs allow to create database copies / instances, conduct resource modification which is available for database copies, delete database copies, create database snapshots (which are similar to backups) and recover database from snapshots [5].

However, the modern approach of using cloud database service API in cyber-physical systems is not completely worked out. The main problem is that an API request could last for a long period. In addition, data replication technology is not fully studied. In many cases, this problem has bad effect on application tasks performance in cyber-physical systems. Certain drawbacks could be avoided by minimizing the service time of API requests from CPhS to CDB.

III. RESEARCH GOAL

The goal of this research is to find a method in order to minimize service time of API requests from a cyber-physical system into a cloud database.

IV. RESULTS OF THE STUDY

Accelerated service time of API requests to cloud database management systems is the most perspective in terms of tracking changes in a local database, change synchronization on cloud database or vice versa.

One of the most popular mechanisms for synchronizing objects copy content is replication. Changes that are made in one copy of the object could be spread into other copies by replication [6]. There are two types of replication: synchronous and asynchronous.

In a case of synchronous replication, if the given replication is updated, all other replicas of the same data fragment should be updated in the same transaction. Logically, that means that there is only one version of data [7]. In most of the products, synchronous replication is performed using trigger procedures (perhaps, hidden and managed by the system). However, one of the disadvantages of synchronous replication is that the system creates significant additional load while performing transactions, where any replicas are updated. (In addition, certain problems such as data access could arise).

With asynchronous replication, an update of one replica has spread to others through a certain amount of

time instead of one transaction. Also, in case of asynchronous replication, there is either delay or waiting time during which separate replicas could be actually nonidentical (as we don't have to deal with exact and timely created copies, to define replica is not relevant) [7].

Based on the analysis of features given above, we could establish that minimization of the request of API service time from CPhS to the CDB management systems is achieved with asynchronous replications.

Therefore, it is relevant to consider the corresponding algorithm.

One of the main elements of asynchronous replication is a primary key, which is mainly zipped with 64-bit binary code (NUMERIC (18, 0)). This is a composite key recorded in one field. There are several approaches, which provide the uniqueness of the primary key. One of the popular solutions are methods which are established on the conversion of keys (the same record has a different primary key in different DB). However, conversion of the key complicates the algorithm and needs an additional amount of time.

The easiest solution is to use timestamps in order to register changes. However, timestamps have certain disadvantages:

- It is hard to set the same time on all of the CDB nodes.
- System time depends on the human factor.
- Various time zones.
- Different ways of time combination in a computer; there is a different idea of time with different accuracy and limitation in every OS, DBMS and programming language.

From all the things stated above, it is recommended to avoid using timestamp and use a logical clock (which is also called generation counter) instead. The value of the logical clock increases every time after each event (in this case the event is a replacement). It is possible to restore the order of changes in one DB. However, it is not possible to restore the order of changes in CDB because you cannot compare the value of the logical clock in different DB. The difference between the generation counter and the logical clock is that generations increase in common cases after several changes. Then these changes belong to one generation.

Service tables (spreadsheets) or so-called register tables (logs) are used in order to register changes in algorithms that have certain properties:

− Each table XXXX (XXXX – code data table) with data has one related registration table LOG_XXXX.

− Each record in the registration table has one related record in data table if the record was inserted or changed.

− If the record was deleted, then the records from the registrar are not related to any record from the table.

The ID of records in the data table is an external key, which refers to the related record from the registrar table [7]. Registrar table LOG_XXXX has certain structure shown in Table 1.

The client will call the general features of the replication algorithm that receives data. The server will call database. Replication starts from client initiative.

Table 1

Data registration table structure

Purpose	Physical name
Record ID	LOG_GID
Insert generation	LOG_INS_GEN
Update / Delete generation	LOG_UPD_GEN
Sign of deletion	LOG_IS_DELETED
ID DB record owner (DB where the record was created)	LOG_DB_ID

The sequence of data transfer can be separated into several steps:

1. Request of the current state of the server generation counter (Server.CURR_GEN).

2. For each generation, starting from the given server generation +1 (Choose from DB_Profile) to Server.CURR_GEN (changes are unknown for the client) execute:

2.1. Start transaction; install replication condition (Client.BEGIN_RECEIVE).

2.2. Receive changes from server (Server_DB-ID-GEN, Server_MASTER-GEN, Server. DETAIL_GEN); changes are memorized in buffer tables (Client.DB_PROFILE_INPUT. Client.MASTER_INPUT, Client.DETAIL_INPUT).

2.3. Apply update changes (Client.PROCESS_ INPUT_TABLES): (main-subordinate sequence); insertion (main-subordinate sequence), deletion (subordinate-main sequence).

2.4. Disable state of replication, memorize server generation (Client.END_RECEIVE); approve the transaction.

2.5. Set new generation (Client.SET_NEW_GEN); installation of new generation performed in a special transaction.

In the case of an exceptional situation, replication could be prolonged from the last successfully accepted generation instead of the beginning. During the realization, the process server memorizes which generation the client has accepted (Server.SET_ CLIENT_ GEN). It is essential for clearing logs from "dead" records. Apart from the Server.SET_ CLIENT_GEN data are transferred only from the server to the client.

Accelerated service of the API requests to the database management system could be set based on algorithm modification by providing background data synchronization in DB and CDB management. The algorithmic-programming realization principles of this approach are shown in the example of the implementation of versatile data integration between Salesforce cloud database management and Ruby on Rails application. It is expedient to show this software tool in a gem (library package) [6] aspect for practical use.

The authors suggest an approach that involves using local storage in the form of any relational database, using object-relational mapping, which makes it easier to read and modify data in database tables and synchronization between the remote cloud database as a whole. Also, this approach involves data processing in the background mode, without user intervention in the synchronization process.

An important aspect of the proposed approach is to maintain the integrity and synchronization of both data storage. Accelerated service method of API requests to the cloud database management based on the application of a synchronous accumulative table. The primary step would be the formulation of the table, which contains all important information about the exact changes which happened in local and remote databases from the last synchronization. Conclusion based on the analysis of the given table is made in both local and remote databases. The conclusion identifies what exact record needs to be updated, pasted or deleted. Accordingly, synchronization methods should consist of three main modules: Accumulative table filter, Accumulative table analyzer, and API call executor, which are shown in Fig. 1.

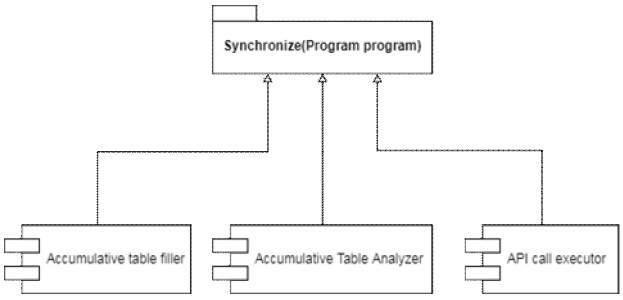

Fig. 1. The main components of API service requests from CPhS to CDB

These modules are independent of each other, so as you continue to work on a project that includes API requests from cyber-physical system to cloud database, it is possible to modify each of three parts separately with no effect on other modules to improve results as the given parts are functionally independent of each other. It is proper to realize these parts as modules, as it allows to use them in various applications such as web sites, desktop and even mobile apps.

In addition, each module includes several submodules.

The data storage module called as Accumulative Table Filter includes the following sub-modules (Fig. 2):

- Timestamp file reader;
- Search synchronous models;
- Module to execute API requests for changes that occurred in the remote database.

Accumulative table analyzer contains (Fig. 3):

- Parser collecting changes to the local repository;
- Parser collecting changes in the remote repository;
- Local storage synchronization module;
- Module for comparing local changes and changes to the remote database.

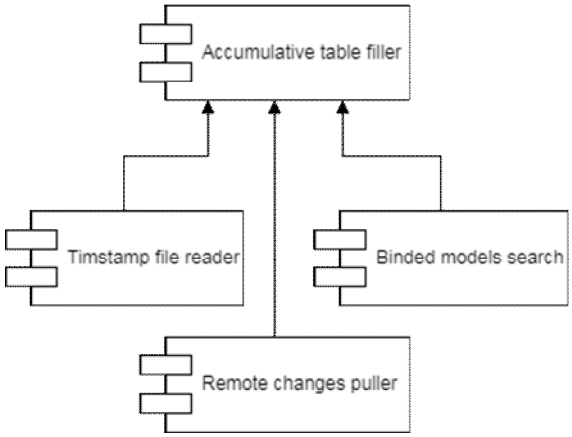

Fig. 2. The components of Accumulative Table Filter

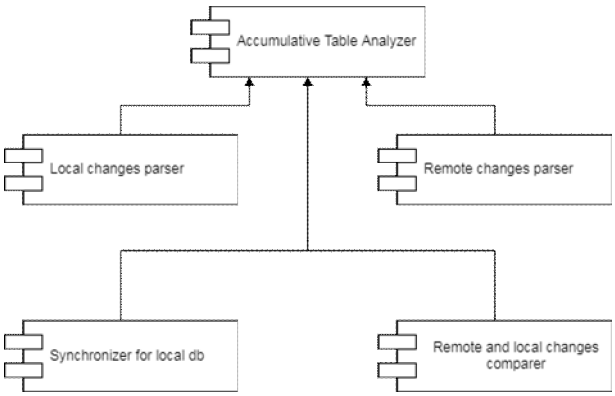

Fig. 3. The components of Accumulative Table Analyzer

The module for API request execution consists of (Fig. 4):

- Module for executing Update calls;
- Module for executing Insert calls.

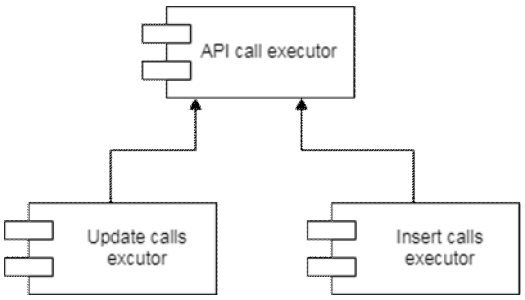

Fig. 4. The components of API call Executor

Suggested module library that related to minimization of API service request could be implemented in any Ruby on Rails application. A developer has an opportunity to link any model (in a case that model matches the look of remote table) to synchronization with the remote table in the cloud database.

The person who admins the cloud database management system should be able to modify the data of the remote tables. As a result, the table in the local database must acquire the same status as the remote one.

According to the research results, the software tool was implemented with several functions. First, the system can create a communication channel between local database, Ruby on Rails application and cloud database on the Salesforce platform, based on configuration file with clearly defined coordinates of endpoints. Secondly, one of the functions is the ability to set various synchronization strategies (constant, passive, associative). Another function is also the ability to synchronize separate fields and associative relationships in the database. User has an ability to specify separate fields in a table for synchronization and type of associative connection with other tables in a database by describing synchronization logic in relational models. The system analyzes description and performs synchronization of the associative tables and separate fields. All synchronization events are proceeding asynchronously in a separate flow that provides a higher level of service comparing to direct API request.

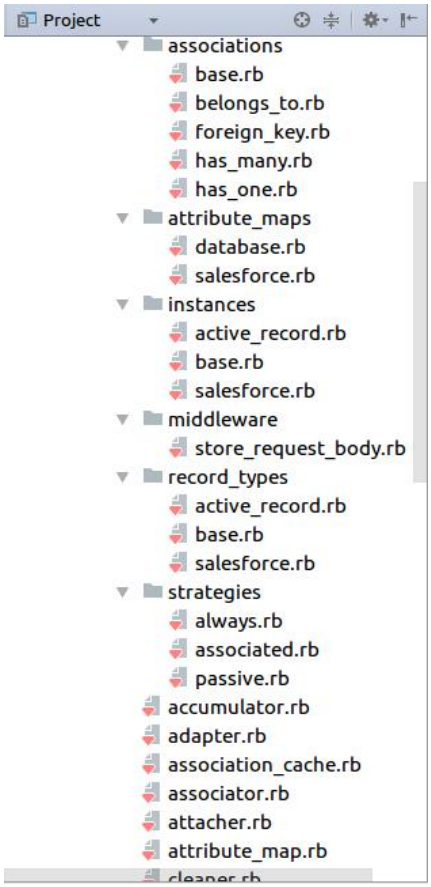

Fig. 5. The list of modules on the gem library

The authors used Ruby (programming language) for the development of the test software for data synchronization between the local database and cloud database on the Salesforce platform. Rails was used as a software frame, and PostgreSQL for saving data. Software is implemented in a library form (gem) for universal use in any Ruby on Rails application. The list of implemented modules is given in Fig. 5. The package manager is also used to ease the creation process, distribution and library installation (distributed package manager) [8], [9].

These modules set logical conditions of library functional modes:

1) Restforce::DB::Associations::Base – sets an associative link between two mappings in a mapping register.

2) Restforce::DB::Associations::BelongsTo – determines connection of record reaction to specific table in SalesForce database.

3) Restforce::DB::Associations::ForeignKey – determines salesforce_id as an external key.

4) Restforce::DB::Associations::HasMany – determines affiliation link of various Salesforce records to a certain table in a SalesForce database.

5) Restforce::DB::Associations::HasOne – determines an affiliation link of one SalesForce records to a certain table in a SalesForce database.

6) Restforce::DB::Instances::Salesforce – serves as a wrap for SalesForce objects, giving common API for record attribute regulation with ActiveRecord.

7) Restforce::DB::Middleware::StoreRequestBody – provides storage of the body request in the environment with HTTP-client.

8) Restforce::DB::RecordTypes::Salesforce – serves as a wrap for only Salesforce object class which allows to make an overview of the attributes and mapping of attributes.

9) Restforce::DB::Strategis::Always – defines initiation strategy for mapping, where new-founded records should be always synchronized with Salesforce in the database and vice versa.

10) Restforce::DB::Strategies::Associated – defines initiation and mapping strategy, in which new-founded records should be synchronized in another system, but only that case, when the certain associative record will be synchronized.

11) Restforce::DB::Strategies::Passive – defines initiation strategy for mapping where new-founded records should not be synchronized in another system.

12) Restforce::DB::Worker – represents a continuous polling cycle through which all record synchronization takes place.

13) Restforce::DB::Syncronizer – relates to record synchronization in Salesforce with the records in the local database.

14) Restforce::DB::Registry – tracks all mappings which fall into its registrar.

15) Restforce::DB::Railtie – allows to start necessary Rake:Tasks in every Rails application.

16) Restforce::DB::Mapping – covers multiple reflections between database columns and Salesforce field by providing utilities for converting hash attributes from one to another.

17) Restforce::DB::DSL – determines syntax that helps to set reflection between the database model and other types of objects in Salesforce.

18) Restforce::DB::Configuration – provides methods of rectilinear reading and recording in order to allow users to set Restforce:: DB.

19) Restforce::DB::Accumulation – refers to the accumulation of changes during one synchronization run.

The approach refers to time minimization of serving API request, which is realized by the created library, provides usage of local storage on the basis of any relational database, usage of ORM [8], which simplifies reading and modification data in database tables and synchronization between remote cloud database in general.

The algorithm of the application could be split into several linear and nonlinear scenarios.

The work of application started through corresponding cycle integration of which is performing data synchronization. The cycle is processing on the background mode without direct communication with the user.

Further processes occurred during the iteration cycle:

1) Reading of the timestamp file. This file includes data that show the last successful synchronization.

2) Search of connected models. Special DSL [8] shows which ActiveRecord ORM models and their field could be synchronized.

3) Diff file formulation is based on changes in records in the local and remote database and timestamp data.

4) Two-side synchronization.

The data synchronization process is based on the step-by-step review of the diff file and on the performance of such commands like insert, update, delete in relation to the local and remote databases.

The generated Diff file is based on changes, which happened on the local and remote databases. That is, conditionally, file could be divided into two sections: local_diff and remote_diff. Changes that occurred in the local database are stored in the first section, and changes that happened in the remote database. Deleting data from the local database is one of the important factors as the process will be recorded into local_diff. Otherwise, using timestamp these changes would be impossible to track.

The synchronization process is given through the related algorithm. There are a few important features in this algorithm. Collector.Run – a method running of which launches step by step data gathering process that was modified both in local and remote databases. Timestamp.Cache.timestamp receives the most recent saved timestamp for a transmitted object. Timestamp falls into marked time slots to make sure that this cycle informed about modifications, which were made during the previous run. Synchronizer run synchronizes records for current comparison between changes in the local and remote databases. This algorithm relies on connections that are present in models subjected to synchronization.

Evaluation of benefit regarding method application of accelerated API service based on test cases and on the usage of ten typical queries to the database management system. These queries are received using the Active Record ORM tool [8], [10]. Actual time is compared using synchronous and asynchronous requests to get a quality estimate of the proposed improvements. Quantitative values of related and absolute performance are also being calculated.

CRUD (create, read, update, delete) are basic operations used in most database management systems. These operations are interpreted in related requests (insert, select, update, delete), which are used by a remote database for direct work.

After library launch, using demonstrative application comparison between usual and asynchronous API requests is conducted. Basic operations performed in every cloud or local database were used in these compartments. Comparison of performance conducted for the basic operations is given in Figs. 5–8.

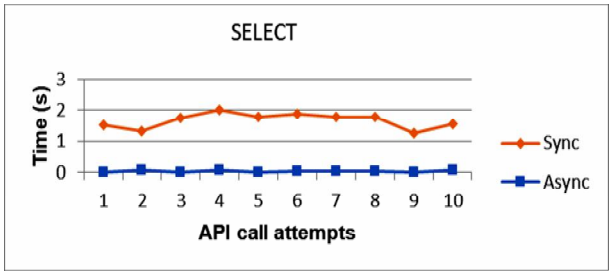

Fig. 6. Page rendering time dependency for read operation

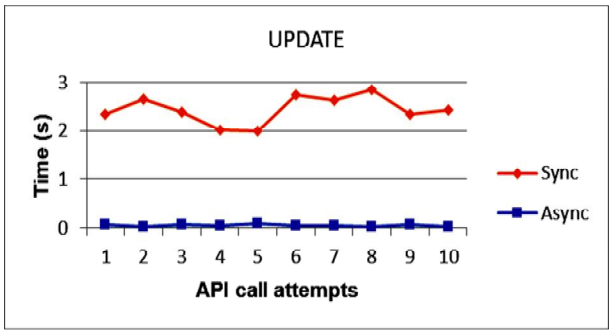

Fig. 7. Page rendering time dependency for update operation

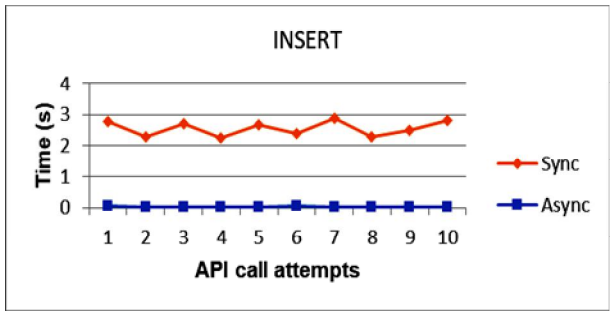

Fig. 8. Page rendering time dependency for create operation

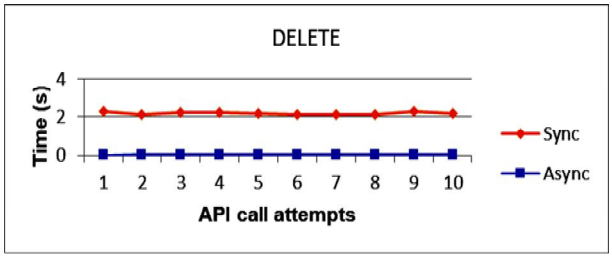

Fig. 9. Page rendering time dependency for Delete operation

Received results of the test cases approve service time minimization of API requests from cyber-physical systems to database management systems with the provided asynchronous method.

V. CONCLUSION

Results of modern state of CPhS analysis showed the necessity of cloud calculation usage. The work necessity of CPhS database and cloud database was set. In addition, expediency of time service minimization of API requests was shown. Service time of API requests highly depends on features of replication. It proved that generation counter data usage is more effective in the replication process rather than a system timer. The asynchronous minimization method of service API requests from CPhS to cloud database systems is proposed and investigated. The method is based on applying of synchronous accumulation table and changing registration in a database using the two-step generation setting. The implemented library provides execution of asynchronous API requests from CPhS to Salesforce database management system.

The possibility of any Ruby on Rails application is foreseen in the library. The effectiveness of the proposed solutions and service time minimization of API requests from CPhS to cloud database by the proposed method was checked on the test cases.

REFERENCES

[1] Melnyk A. Cyber-physical systems: problems of creation and directions in development. *Transactions on Computer systems and networks, Lviv Polytechnic National University Press,* No. 806, 2014, pp. 154–161 (in Ukrainian).

[2] Chappell D. A Short Introduction to Cloud Platforms and Enterprise – Oriented View, Chappell and Associates, San Francisco, 2008, pp. 1–13.

[3] Lee G. Cloud Computing: Principles, Systems and Applications / Nick Antonopoulos, Lee Gillam. L.: Springer, 2010, pp. 23–24.

[4] Qusay H. Demystifying Cloud Computing / Hassan, Qusay. – *The Journal of Defense Software Engineering. CrossTalk*, 2011, pp. 16–21.

[5] Galen G. What cloud computing really means. [Online]. Available: https://en.wikipedia.org/wiki/InfoWorld

[6] Fowler M. Patterns of enterprise application architecture. Addison-Wesley. 2015, 47 p.

[7] Pavych N., Kutkovyi B. Accelerated servicing method of API calls to cloud-database management systems. *Transactions on Computer systems and networks, Lviv Polytechnic National University Press*, No. 883, 2017, pp. 154–161 (in Ukrainian).

[8] Gilmore J. Meet DataMapper ORM, the Unified Ruby Interface for Data Stores. [Online]. Available: https://www.developer.com/lang/rubyrails/meet-datamapper-orm-the-unified-ruby-interface-for-data-stores.html

[9] Rowe M. The POSTGRES data model, *Proceedings of the 13th International Conference on Very Large Data Bases*, Brighton, England: Morgan Kaufmann Publishers, 2016, pp. 83–96.

[10] Kutkovyi B., Pavych N. (2017). API-calls optimization for cloud database management systems. Pres. at International Scientific Journal "Internauka", No 14, 2017. [Online]. Available: https://www.internauka.com/en/issues/2017/14/3003.

Nataliya Pavych. Ph.D., Assoc. Prof at Software Department of Lviv Polytechnic National University. Master degree of Applied Mathematics in 1996 at Ivan Franko National University of Lviv. Her Ph.D. thesis was defended in 2001. Research in the database, data warehouse, data analysis areas. Author of more than 50 scientific works.

Tetyana Pavych. Junior Business Analyst at Titan International. Bachelor's degree in Management Information System in 2019 at Northwood University, Midland, MI, USA. Research in the CRM management, business intelligence, data analysis area.