information

Article

Data Processing in Cloud Computing Model on the Example of Salesforce Cloud

Witold Marańda [1], Aneta Poniszewska-Marańda [2,*,†] and Małgorzata Szymczyńska [2]

[1] Department of Microelectronics and Computer Science, Lodz University of Technology, 93-590 Łódź, Poland; witold.maranda@p.lodz.pl
[2] Institute of Information Technology, Lodz University of Technology, 93-590 Łódź, Poland; 220095@edu.p.lodz.pl
* Correspondence: aneta.poniszewska-maranda@p.lodz.pl
† Current address: al. Politechniki 8, 93-590 Łódź, Poland.

Abstract: Data processing is integrated with every aspect of operation enterprises—from accounting to marketing and communication internal and control of production processes. The best place to store the information is a properly prepared data center. There are a lot of providers of cloud computing and methods of data storage and processing. Every business must do the right thing, which is to think over how the data at your disposal are to be managed. The main purpose of this paper is research and the comparison of available methods of data processing and storage outside the enterprise in the cloud computing model. The cloud in SaaS (software as a service) model—Salesforce.com and a free platform development offered by Salesforce.com—force.com were used to perform the research. The paper presents the analysis results of available methods of processing and storing data outside the enterprise in the cloud computing model on the example of Salesforce cloud. Salesforce.com offers several benefits, but each service provider offers different services, systems, products, and forms of data protection. The choice of customer depends on individual needs and business plans for the future. A comparison of available methods of data processing and storage outside the enterprise in the cloud computing model was presented. On the basis of collected results, it was determined for what purposes the data processing methods available on the platform are suitable and how they can meet the needs of enterprises.

Keywords: cloud computing; data processing; information systems development; data storage; computational methods; Salesforce cloud

Citation: Marańda, W.; Poniszewska-Marańda, A.; Szymczyńska, M. Data Processing in Cloud Computing Model on the Example of Salesforce Cloud. *Information* **2022**, *13*, 85. https://doi.org/10.3390/info13020085

Academic Editor: Vincenzo Moscato

Received: 3 January 2022
Accepted: 9 February 2022
Published: 12 February 2022

1. Introduction

Data processing is integrated with every aspect of operation enterprises—from accounting to marketing and communication internal and control of production processes. Businesses are struggling these days to deal with the increasing amount of information collected and processed. This prompts them to pay extra attention toward making the best use of the available space data storage and optimization of stored information based on scrupulous analysis of company needs [1]. Data processing is essential and important for day-to-day operations as well as for long-term planning strategic for most companies. Large amounts of collected data can be used to find long-term business opportunities and for monitoring consumer trends that may have an impact on different aspects inputs, quotes, or company products. Appropriate information processing and analysis are irreplaceable in obtaining results that support making the process decisions leading to the solution of problems arising in the enterprise [1–3].

In the ever-evolving world of new technologies, data are produced in enormous amounts, and the more information is collected, the more difficult it is to manage. It is also more and more difficult to carry out the process of transforming raw data into meaningful information that could improve the current situation or solve existing problems of the

enterprise. Large amounts of data should also be properly stored and protected against damage or theft [4].

The best place to store the information is a properly prepared data center with a properly equipped and secured server room. Maintaining such a server room within the enterprise is not an easy task. The company must ensure constant monitoring of the operating condition of devices and installations, as well as conditions prevailing in the server room and proper maintenance of the entire system aimed at excluding any breakdowns and access to the interior by unauthorized persons [5]. Due to the fact that the amount of information stored in companies increases every year, problems arise related to the lack of equipment and space needed for data storage. It can also be troublesome to hire and train employees who take care of the appropriate conditions and the security of the data storage room.

As a result, companies were forced to look for ways minimizing costs related to data processing and storage [6–8]. The most popular solution today is the use of cloud computing offered by external service providers. Cloud computing significantly eliminates the problems related to running enterprises' own data centers and helps to optimize storage costs and information processing [3,9–11]. There are a lot of providers of cloud computing and methods of data storage and processing. Every business must do the right thing: think over how the data at your disposal are managed.

Salesforce.com cloud computing offers a lot. However, it cannot be said that it is a cloud better than the rest on the market today. Each service provider offers different services, different systems, different products, and forms of data protection. Which supplier the customer chooses depends on their individual needs and business plans for the future.

The paper's main purpose is to present the analysis results of the available methods of processing and storing data outside the enterprise in the cloud computing model on the example of the Salesforce cloud.

The main contribution of this paper is as follows:

1. Research and comparison of available methods of data processing and storage outside the enterprise in the cloud computing model. The cloud in SaaS (software as a service) model—*Salesforce.com* and a free development platform offered by Salesforce.com—*force.com* were used to perform the research.
2. On the platform, selected data processing methods were implemented. For each method, factors such as the duration of a given transaction and the number of cancellations were collected and compared to the database, the number of processed records in the given time, concurrency, and the degree of parallelism.
3. Based on the collected results, it was determined for what purposes the data processing methods available on the platform are suitable and how they can meet the needs of enterprises.

The paper is structured as follows: Section 2 presents the overview of data processing and storage together with the problems related to them. Section 3 describes the related works on cloud computing data storing and processing. Section 4 presents the selected aspects of data processing in Salesforce platform, while Section 5 deals with the analysis of data processing methods delivered by Salesforces cloud. It also determines for what purposes the selected methods of data processing can be used by businesses. The last section concludes the paper together with the discussion of presented results.

2. Data Storing and Processing

The definition of business data processing covers all possible operations performed on data. The most common are the modification, transmission, analysis, management, and collection of data. Transformation is the strategic goal of performing data operations raw data into meaningful information that helps improve the current one the situation in the company or solve an existing business problem. The output processing often takes many forms, such as reports, diagrams, and graphics, making the data easier to understand and analyze [1,2].

Data processing is the process responsible for the conversion content and input through systematic execution operations to obtain the output in a predetermined form [4,12]. The following data processing methods are distinguished [4]: manual data processing, mechanical data processing, and electronic data processing. Electronic data processing can be further divided due into processing techniques:

Batch Processing—gathers data into groups, which later remain processed sequentially in the correct order. The processing of subsequent groups data does not require human intervention. The processes start automatically one after the second. The first process starts at a predefined time, most often when processing systems are as little loaded as possible with other tasks. Thanks to this, it is possible to perform operations on many records in a fairly short time. The processing results are later saved to the database in the format, enabling their further use (displaying data in the system, analysis, creating documents, and reports).

Real-Time Processing—the processing that is used to perform operations on real-time data. It is used when the processing results must be displayed in the shortest possible time. Data provided to the software are used immediately.

Multiprocessing—this type of processing provides the basis of all processor-based computing devices. It consists of the task or sets of operations being shared among the processors working together. This type of processing reduces the overall execution time all operations and increases the efficiency of the process. In addition, the available processors operate independently of each other. This means that the failure of one processor does not cause stopping the entire process as other processors continue to run.

Time-Sharing Processing—this is the processing in which a single processor is used by many users. Performing the given operations takes place in different time intervals for different users as allocated by the processor sometimes. Switching occurs so frequently that any user can expect a response very quickly.

The data processing process is cyclical. The data processing cycle is a series of steps performed to extract the relevant information. Changes on the market, new customers, and changes in the enterprise lead to the need to repeat the data collection stage, which results in the next data processing cycle. The processing cycle shows how data change their form, facilitating its further interpretation and allowing the final results to be used in effective business decisions [1].

Problems Related to Data Storage and Processing

Data storage is an extremely demanding and costly process, which brings with it many problems. In today's data-intensive world, many companies focus on analytics and processing the information they collect. The main problem becomes what to do with all the collected data. The more data a company has, the more difficult it is to manage. The most important problems related to the processing and storage of large amounts of data include [2]:

Provision of appropriate infrastructure—various types of confidential information, which is important from the company's point of view, must be stored and processed in professionally prepared conditions that ensure adequate security in every aspect [4]. The method of access to information and the rules for handling data should be clearly defined.

Storage and processing costs—the purchase of servers, employee training, employment of new specialists, and ensuring security and appropriate physical and technical conditions require considerable financial outlays. This is a significant problem related to the storage of large amounts of data in the company.

Data security—it is very difficult to ensure adequate data security in the company. Unfortunately, even those companies that spend enormous amounts of money on protection face problems. The most famous organizations whose data came into the hands of cybercriminals include Sony PlayStation Network, Allegro, Filmweb, and Facebook. Companies that lead to such situations have to deal with legal ramifications later. Unfortunately, both the security inside the company and the security offered by cloud providers do not

guarantee full protection. In internal data centers, the company is responsible for its own security. By using the cloud, enterprises entrust their data to an external provider who may not have the most up-to-date security certificates. Cloud service providers store customer data backups in various locations around the world. Each new location reduces the risk of data loss but unfortunately also increases the risk of their being stolen [12].

Data collection—as data are usually collected from several data sources, duplication occurs many times. The same records may appear more than once for various reasons. Redundancy causes problems when processing and analyzing data. Reports and calculation results often turn out to be incorrect, and all operations have to be repeated, wasting a lot of time. When large amounts of data are collected, there is no guarantee that the data are consistent, complete, or correct.

Amount and compatibility of information—companies are constantly developing and therefore collecting more and more data. The initially selected data storage location may not be large enough after some time. Future needs cannot be predicted with precision. It is therefore important to ensure the appropriate scalability of the potential data center. Data processing applications that functioned well for a small amount of data may not work the same when the amount of information increases. Thus, it is important to find out which historical data would still be useful and which can already be disposed of. Another problem is the issue of data compatibility. The more data an enterprise collects, the more information processing applications it usually uses. Ensuring that the format of all data in all used applications is consistent is a very complicated task that requires the involvement of qualified employees and time.

The data storage and processing problems described above are not the only ones that businesses may encounter. Everything cannot be eliminated, but it is definitely worth the time to reduce the risk of high costs of repairing errors. Consider carefully how you can collect, store, and manipulate data between different applications [2].

3. Related Work on Cloud Computing Data Storing and Processing

Cloud computing is a model that allows convenient on-demand access to a common pool configurable computing resources (e.g., for networks, servers, memory mass, applications, and services) that can be quickly made available and released with minimal effort on the part of the service provider [7,8,10,13–15].

Cloud computing providers and the services they offer can be compared on several different levels. The offers of service providers differ significantly depending on the model and type of cloud. There are vendors that specialize in providing infrastructure and some that focus on delivering professional systems and software. Some try to meet the needs of each cloud computing model.

Cloud computing providers and the services they offer can compare on several different levels. This market is dominated by cloud computing giants such as AWS (Amazon Web Services), Microsoft Azure, Google Cloud Platform, IBM Cloud, or Salesforce.com.

The cloud computing study [7,8] shows that more and more organizations are using the cloud at a certain level, along with the growing popularity of public and private clouds. Most companies combine different cloud types at the same time. There are many cloud providers, but the market leaders are still AWS, Amazon, and Google Cloud. The cost of cloud adaptation is the biggest challenge for small- and medium-sized enterprises. It is therefore important to consider for what purposes investing in the cloud is necessary. Often, in order to optimize costs, it is recommended to use the advice of specialists who are able to indicate the appropriate adaptation tactics for a given company. Another important issue is data security in the cloud. Usually, it is not certain where the data are exactly and what people have access to them. Contrary to appearances, resources stored with an external supplier are often more secure than those stored inside a given enterprise.

However, there are also many works on computing data storing and processing in the literature.

The authors of [16] propose the mobile cloud computing model to use the open-source codes from distributed computing frameworks, such as Hadoop. It is defined to improve the efficiency of business processing. They also study how to process and analyze the unstructured data in parallel to this model and also verify if customized information for individuals may be provided using unstructured data.

Muniswamaiah et al. present the review of opportunities and challenges of transforming big data using cloud computing resources [17]. They show how big data are used in decision making process to gain useful outcomes from the data for business and engineering [17]. Presenting the challenges of processing, they evaluate if cloud computing is helpful in advancement of big data by providing computational, networking, and storage capacity.

Kaplancali and Akyol analyze the performance evaluation of small- and mid-sized enterprises (SMEs) form the point of view of cloud computing usage in their activities. The authors conducted the quantitative study on the set of 112 respondents employed in Turkish SMEs. The performance specific scales is used for research model, and the obtained results of signified cloud technology have the positive impact on business performance.

The authors of [18] discussed the cloud computing architecture and its numerous services as well as several security issues in cloud computing based on its service layer. Moreover, several open challenges of cloud computing adoption and its future implications were identified together with the presentation of available platforms in the current era for cloud research and development.

De Donno et al. analyze the foundations and evolution of computing paradigms [19]. They present the evolution of modern computing paradigms and, for each paradigm, show its key points and its relation with the others. The authors address the fog computing and its role as the connector between IoT, cloud computing, and edge computing. Moreover, they identify the open challenges and future research directions for IoT, cloud computing edge computing, and fog computing.

Alkasem et al. present the proposal of a new methodology to construct the performance optimizing model for optimizing the real-time monitoring of the big datasets [20]. This model includes a machine learning algorithms and Apache Spark Streaming to realize the fine-grained fault diagnosis and repair of big dataset. The authors studied the case of use of the failure of virtual machines (VMs) to start up. The proposed methodology ensures that the most sensible action is realized during the procedure of fine-grained monitoring and generates the efficacy and cost-saving fault repair. This process is performed by three control steps: data collection, analysis engine, and decision engine.

The authors of [15] present the recent contributions and results in the fields of cloud computing, IoT, and big data technologies and applications. They specify different concepts of cloud computing technologies from chosen points of view for industrial and medical applications.

Forestiero et al. [21] propose the hierarchical approach for workload management in distributed data centers. Its aim is to preserve the autonomy of single data centers and allow the integrated management of heterogeneous platforms. The described solution is rather generic but, according to the authors, answers the specific requirements of single environments—it is shown by the analysis of performance of specific cloud infrastructure composed of four data centers.

Moreover, the edge cloud computing technology is related to data processing in the enterprises. The authors of [22] describe the mobile edge computing as a new technology that enables the innovative service scenarios to ensure the optimized network operation and new business opportunities. Mobile edge computing opens up services for the consumers and enterprise and also adjacent industries to deliver critical applications and data over the mobile network. Such solution can support the new value chain and cases across multiple sectors.

Lee et al. [23] in the framework of fog computing concept propose the mobile personal multiaccess edge computing (MEC) architecture that utilizes users mobile device as MEC

server (MECS) to allow the mobile users to receive the continuous service delivery. The results shown in the work by the form of proposed scheme reduce the average service delay and provides efficient task offloading compared to the other existing MEC schemes.

The work of Wang et al. [24] presents the edge-based auditing method that is addressed to the data security in the framework of internet of things resources. It proposes the audit model that is based on binary tree assisted by edge computing to provide the computing capability for resource-constrained devices. The data preprocessing task by offloading to the edge gives the possibility to reduce the computing load and improves the processing efficiency.

The authors of [25] describe the concept of multiaccess edge computing (MEC) that gives the computing power and storage resources to the edge of mobile network. In this way, it allows the mobile user device to run the real-time applications. The proposed MEC-based mobility management scheme to arrange MEC server (MECS) enables one to receive content and use server resources efficiently even when they move.

More and more companies and organizations are using the cloud with the growing popularity of public and private clouds. There are many cloud providers, but the market leaders are still AWS, Amazon, and Google Cloud. The cost of cloud adaptation is the biggest challenge for small- and medium-sized enterprises. It is therefore important to consider for what purposes investing in the cloud is necessary. Another important issue is data security in the cloud. Usually, it is not certain where the data are exactly and what people have access to it. Contrary to appearances, resources stored with an external supplier are often more secure than those stored inside a given company. More and more companies recognize and appreciate the advantages of cloud computing, which is why we can observe an increase in its use on the market.

4. Data Processing Using the Salesforce Cloud

Salesforce is a product built on *force.com* by a company called Salesforce.com. The force.com platform is PaaS service that is completely cloud-based. The user does not need the complicated installation of additional software to use it. The force.com platform allows one to create their own applications by using and the extension of existing infrastructure. Because some Salesforce.com are not configurable but run on force.com, they are often perceived as products in the SaaS model. Custom solutions created by developers on the platform, however, are understood as a PaaS model. Salesforce.com is not a typical cloud computing provider. This company does not primarily focus on offering infrastructure (IaaS), but providing business development supports configurable applications according to customer requirement [1].

Force.com is a developer platform provided by Salesforce.com, allowing one to modify the default features offered by this company. Developers can create applications and websites via the integrated environment in the cloud and deploy them quickly on force.com servers. Salesforce.com offers a huge number of built-in functions as well as standard objects that can be used by developers during implementation of new applications. The force.com platform has its own dedicated programming language (Apex), database query language (SOQL, Salesforce object query language), search language, which is used to find text in records (SOSL, Salesforce object search language), and a markup language that enables user interface implementation (Visualforce).

Salesforce.com allows one to process data in several different ways. Standard data manipulation language (DML) operations on a database environments, however, have some limits. We can complete during one transaction at the same time only 150 DML operations: that is [26]: operation of inserting new records (Insert), operation of deleting new records (Delete), operations to modify new records (Update), and operations of joining records together (Merge).

Another limitation is that during one transaction, we can perform a DML operation for only 10,000 records. Due to these limits, force.com offers many different methods of processing more data. The most popular of them are: *Batch Apex*—this is a type of data

processing that allows you to run large tasks that modify thousands or even millions of records. Thanks to Batch Apex, developers can process records asynchronously in batches. Class code logic modifier is called once for each batch of records processed. *Future Apex*—is used to run processes in a separate thread, in a later time (when system resources become available). When using standard synchronous processing, all method calls are performed from the same thread. No additional processing may take place, until the process is complete. Future Apex can be used for whatever operation the developer wants to run asynchronously—regardless of main thread. *Apex Queueable*—allows one to add jobs to the queue and monitor them. It is also a kind of asynchronous processing. Apex Queueable is similar to Future Apex but has additional benefits. *Apex Schedulable*—enables processing of specified data sets intervals. This kind of data manipulation is also asynchronous and can be used in conjunction with other previously described methods processing. *Processing by using triggers*—it is standard synchronous data processing invoked by execution DML operations. Triggers process data in groups of 200 records. Each of the data chunk processing processes have their own computing resources and count separately platform limits for them. There are two types of triggers: Before Triggers (used to update or check values records before saving them to the database) and After Triggers (allow access to field values set by system, such as the Record Id field when inserted into the database).

Applications developing on the force.com platform use the MVC (Model-View-Controller) architectural pattern. This is a common pattern that keeps application logic, user interface, and data model separate (Figure 1).

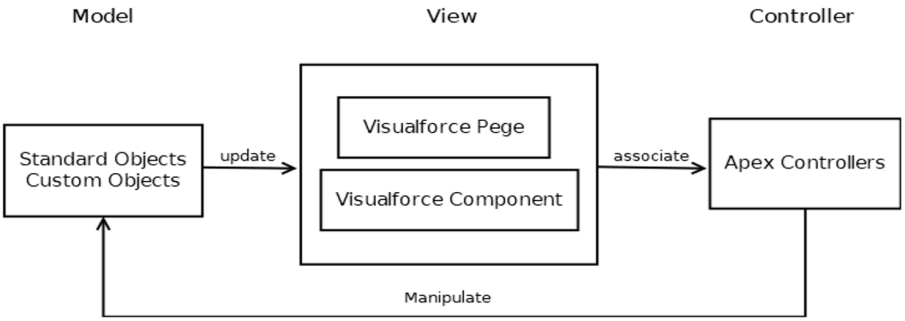

Figure 1. MVC pattern of force.com platform.

The use of MVC pattern in the application *Salesforce Data Processor* created to analyze the Salesforce cloud data processing methods is as follows:

- Model—The Custom Object named *Project_Item_c* was created on the platform. Data processing methods operate on this object. Another created object called *StatisticWrapper_c* was used to collect the statistics of data processing methods in Salesforce cloud, presented in the next section.
- View—one Visualforce page named *DataManipulation.vfp* was created. It allows one to perform data processing operations, add and delete data, and view the results collected during the processing. This page is divided into tabs: Process Data, View Data, and View Statistics Controller.
- Page—two main classes were created: *DataManipulationController* and *DataManipulationHelper*. They contain the application logic. Additionally, classes implementing the *Batchable*, *Schedulable*, *Queueable*, and *Trigger* interfaces were created. This allows the data to be processed using the methods described above.

The Salesforce Data Processor application has three main tabs to make the experiments and analyze the obtained results. The Data Processing tab allows one to select the data processing way (insert, update delete) and to select the data processing method (Figure 2). The View Data tab contains a table that stores the data currently in the system. The Processing Method column represents the last operation that was performed on a given record

(Figure 3). The View Statistics tab shows a two-column table containing statistics data about the last data processing performed (Figure 4).

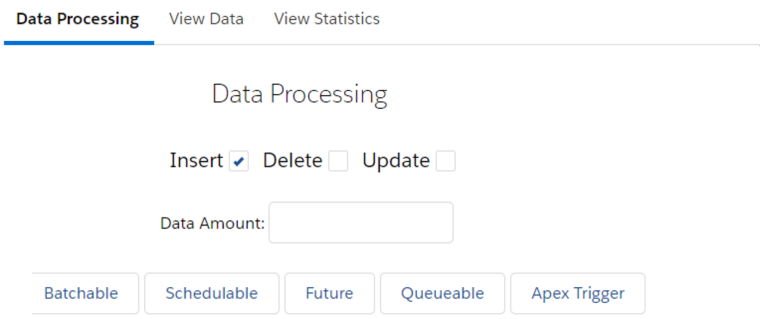

Figure 2. Data Processing tab of *Salesforce Data Processor* application created to analyze the data processing methods.

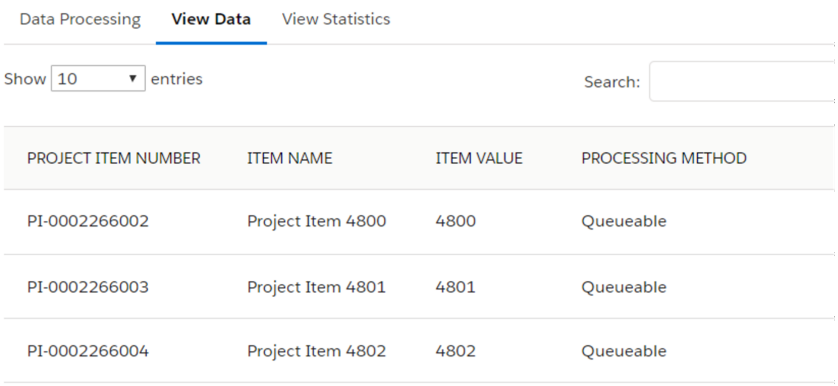

Figure 3. View Data tab of *Salesforce Data Processor* application created to analyse the data processing methods.

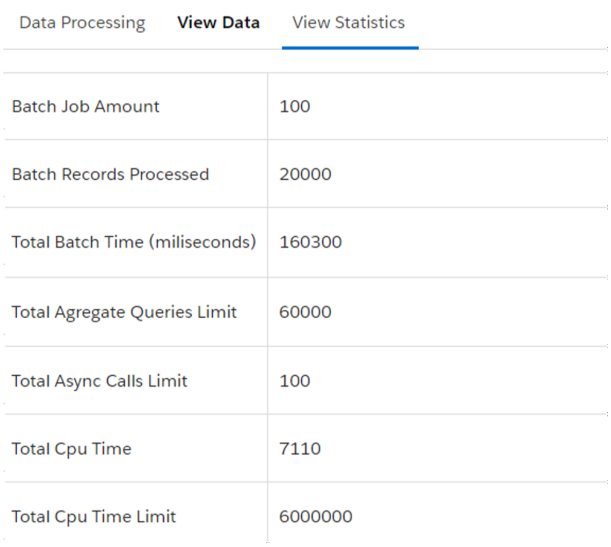

Figure 4. View Statistics tab of *Salesforce data processor* application created to analyze the data processing methods.

5. Analysis of Data Processing Methods in Salesforce Cloud

Each enterprise has its own individual business processes needs and a characteristic way of working. Data processing methods described in the previous section differ in their operation. Any of these methods when used appropriately can contribute to an increase

company performance. Proper data processing supports the achievement of company business goals and optimizes the work of its employees.

Data processing in Salesforce.com cloud was performed for operation of type *insert*, *update*, and *delete*. All these operations were performed for the following methods: *Batch*, *Apex*, *Future Apex*, *Apex Queueable*, *Apex Scheduable*, and *Apex Triggers*. The following factors were compared:

- total duration of a given transaction;
- number of SOQL (*Salesforce Object Query Language*) queries;
- number of DML (*Data Manipulation Language*) operations;
- number of records processed per second, processing time on server (*Central Processing Unit Time*);
- memory used during processing (*Heap Memory*).

Number of 10,000, 50,000, and 100,000 records were inserted into the database, modified, or deleted. The collected results and conclusions are presented below.

5.1. Batch Apex Method

Results were initially collected for the insert operation. The data remained created in the start() method and as a list of *Iterable* objects were passed to execute() method of the *Batchable* interface. The duration of the transaction was counted from the moment calling the *start()* method until the *finish()* method is called. Tests were performed on the data portions divided into 5, 25, 50, 250, and 500 transactions. Standard Batch Apex processes data in groups of 200 records. However, this number can be increased. The maximal number of records for a single transaction is 2000 (Table 1).

Table 1. *Batch Apex* processing results for *insert* operation.

	10,000 records: 5 transactions by 2000 50 transactions by 200	50,000 records: 25 transactions by 2000 250 transactions by 200	100,000 records: 50 transactions by 2000 500 transactions by 200
Total time of transaction duration (seconds)	5/2000: 15.3 50/200: 20.1	25/2000: 81.65 250/200: 182.71	50/2000: 158.29 500/200: 404.2
Number of SOQL queries	In both cases: 0	In both cases: 0	In both cases: 0
Number of DML operations	5/2000: 5 50/200: 50	25/2000: 25 250/200: 250	50/2000: 50 500/200: 500
Number of records processed in 1 s	5/2000: 653 50/200: 498	25/2000: 612 250/200: 274	50/2000: 632 500/200: 247
Processing time on server (seconds)	5/2000: 1.9 50/200: 3.01	25/2000: 10.7 250/200: 24.53	50/2000: 21.24 500/200: 35.61
Memory used during processing (MB)	5/2000: 2.03 50/200: 4.4	25/2000: 10.67 250/200: 71.42	500 transactions: 500/200: 265.09

The more records are processed, the fewer the number of records are processed in one second. It is significantly influenced by the number of transactions, which run on the server. When there are few transactions (5, 25, 50 transactions), the number of records processed per second is similar for all studied cases. Processing the same number of records for more transactions (50, 250, and 500 transactions), it significantly slows down the whole process processing.

For the *delete* and *update* operations, the data was fetched from the database in the *start()* method and passed to the *execute()* method using the *Database.*
QueryLocator object (Tables 2 and 3).

Table 2. *Batch Apex* processing results for *delete* operation.

	10,000 records: 5 transactions by 2000 50 transactions by 200	50,000 records: 25 transactions by 2000 250 transactions by 200	100,000 records: 50 transactions by 2000 500 transactions by 200
Total time of transaction duration (seconds)	5/2000: 24.26 50/200: 31.11	25/2000: 118.21 250/200: 174.8	50/2000: 230.21 500/200: 358.89
Number of SOQL queries	In both cases: 0	In both cases: 0	In both cases: 0
Number of DML operations	5/2000: 5 50/200: 50	25/2000: 25 250/200: 250	50/2000: 50 500/200: 500
Number of records processed in 1 s	5/2000: 412 50/200: 321	25/2000: 423 250/200: 339	50/2000: 434 500/200: 278
Processing time on server (seconds)	5/2000: 0.7 50/200: 1.25	25/2000: 4.01 250/200: 5.78	50/2000: 7.9 500/200: 11.54
Memory used during processing (MB)	5/2000: 2.51 50/200: 3.41	25/2000: 51.2 250/200: 65.92	25/2000: 116.1 500/200: 254.03

The *delete* operation performed on the same number of records as the operation *insert* occurs much faster. The biggest difference is in between processing a large number of transactions for the same number of records. Large amounts of data are deleted almost twice as fast (for 250 and 500 transactions). Compared with the *insert* operation, the data processing time on the server also decreases (Table 2).

The results collected for the *update* operation are similar to the results collected for *insert* operation. However, it can be noticed that for a large number of transactions (50, 250, and 500), there is much less variation in overall process time.

Table 3. *Batch Apex* processing results for *update* operation.

	10,000 records: 5 transactions by 2000 50 transactions by 200	50,000 records: 25 transactions by 2000 250 transactions by 200	100,000 records: 50 transactions by 2000 500 transactions by 200
Total time of transaction duration (seconds)	5/2000: 15.6 50/200: 22.27	25/2000: 86.32 250/200: 111.38	50/2000: 139.72 500/200: 215.44
Number of SOQL queries	In both cases: 0	In both cases: 0	In both cases: 0
Number of DML operations	5/2000: 5 50/200: 50	25/2000: 25 250/200: 250	50/2000: 50 500/200: 500
Number of records processed in 1 s	5/2000: 641 50/200: 449	25/2000: 579 250/200: 448	50/2000: 715 500/200: 464
Processing time on server (seconds)	5/2000: 2.6 50/200: 2.76	25/2000: 13.69 250/200: 13.86	50/2000: 26.21 500/200: 27.01
Memory used during processing (MB)	5/2000: 1.3 50/200: 3.8	25/2000: 66.5 250/200: 67.92	50/2000: 156.98 500/200: 258.02

5.2. Future Apex Method

The results for *insert*, *update*, and *delete* operations were also collected for this method. The duration of the transaction was counted from the moment it was launched asynchronous processing (Tables 4–6). Comparing the processing results for the insert operations of Future Apex and Apex Batchable methods, we can notice that Future Apex is running a little slower. The total time of processing for the same number of transactions and records is greater for this method (Tables 1 and 4). This may be because an asynchronous process is waiting to allocate server resources.

Table 4. *Future Apex* processing results for *insert* operation.

	10,000 records: 1 transactions by 10,000 5 transactions by 2000	50,000 records: 5 transactions by 10,000 25 transactions by 2000	100,000 records: 10 transactions by 10,000 50 transactions by 2000
Total time of transaction duration (seconds)	1/10,000: 25.02 5/2000: 27.35	5/10,000: 139.11 25/2000: 159.50	10/10,000: 342.35 50/2000: 344.59
Number of SOQL queries	In both cases: 0	In both cases: 0	In both cases: 0
Number of DML operations	1/10,000: 1 5/2000: 5	5/10,000: 5 25/2000: 25	10/10,000: 10 50/2000: 50
Number of records processed in 1 s	1/10,000: 399 5/2000: 365	5/10,000: 359 25/2000: 313	10/10,000: 292 50/2000: 290
Processing time on server (seconds)	1/10,000: 2.8 5/2000: 3.1	5/10,000: 10.75 25/2000: 12.21	10/10,000: 22.30 50/2000: 24.45
Memory used during processing (MB)	1/10,000: 1.82 5/2000: 1.89	5/10,000: 94.81 25/2000: 94.95	10/10,000: 189.1 50/2000: 189.96

Table 5. *Future Apex* processing results for *delete* operation.

	10,000 Records
Total time of transaction duration (seconds)	21.95
Number of SOQL queries	1
Number of DML operations	1
Number of records processed in one second	455
Processing time on server (seconds)	0.81
Memory used during processing (MB)	0.62

Tables 5 and 6 do not present the processing results for 50,000 and 100,000 records to update and delete operations. It is caused by the fact that running tasks block each other. Each of the trades we run is independent, and they all start almost in the same moment. The transaction cannot manipulate the records that are in it and are at same time processed by another process.

For 10,000 records, the *delete* operation performed faster for the Future Apex method than for Apex Batchable method. The memory was also smaller in this case used during data processing (Tables 2 and 5). It may be related to only one transaction being run for Future Apex, which has quickly processed all the data. The server resources did not have to be distributed over several different transactions. Due to the locking of records, we cannot do this in the case of performing the tests on more transactions.

Table 6. *Future Apex* processing results for *update* operation.

	10,000 Records:
Total time of transaction duration (seconds)	27.23
Number of SOQL queries	1
Number of DML operations	1
Number of records processed in one second	367
Processing time on server (seconds)	2.97
Memory used during processing (MB)	1.23

Future Apex processing for the *update* operation works similar to Apex Batchable processing for 10,000 records. There are slight differences in total process duration and server load (Tables 3 and 6). Unfortunately, due to the locking of records, it cannot be said that Future Apex is suitable for processing large amounts of data. The maximum number

of records that can be processed in a single transaction is 10,000, and we can only run one transaction for the *delete* and *update* operations.

When the platform limits are exceeded, the administrator is notified of the failure via email. This message specifies the cause of the problem and precisely points to the line of code that caused it. Errors occurring during the process can also be read in the development console logs (Figure 5).

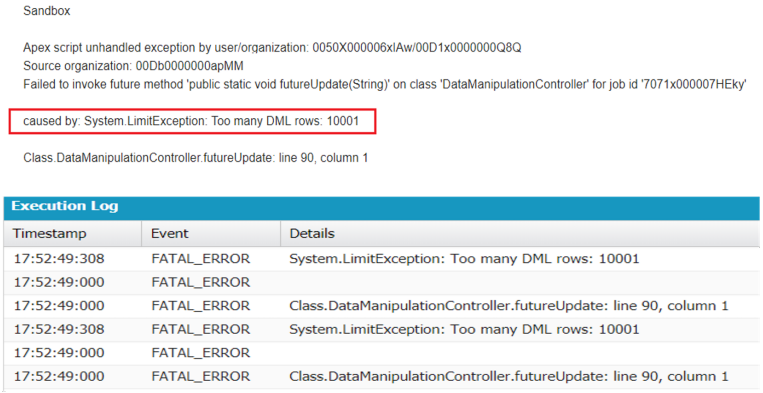

Figure 5. Exceeding the DML operation limit—email notification and collected processing logs in the development console.

5.3. Apex Queueable Method

Processing of this type allows queuing tasks. These tasks are dependent on each other. The collected research results are presented in Tables 7–9.

Total transaction processing time for the insert operation of Apex Queueable method is smaller compared to previously tested methods (Future Apex, Apex Batchable) (Table 7). However, the research did not take into account the time that each queue processing the data had to wait for resources to be allocated. Now, it can detect when a given queue has finished processing. We can only examine the execution time of each individual processing queue.

The research on *insert*, *update*, and *delete* operations was carried out for 1, 5, 10, 25, and 50 record queues. The maximum number of independent queues that the processors can run are 50. During one queue, we can process a maximum of 10,000 records. When trying to use more queues, the platform returns an error.

Table 7. *Apex Queueable* processing results for *insert* operation.

	10,000 records: 5 turns by 2000 1 turn by 10,000	50,000 records: 25 turns by 2000 5 turns by 10,000	100,000 records: 50 turns by 2000 10 turns by 10,000
Total time of transaction duration (seconds)	5/2000: 13.91 1/10,000: 11.32	25/2000: 59.25 5/10,000: 53.51	50/2000: 117.55 10/10,000: 112.98
Number of SOQL queries	In both cases: 0	In both cases: 0	In both cases: 0
Number of DML operations	5/2000: 5 1/10,000: 1	25/2000: 25 5/10,000: 5	50/2000: 50 10/10,000: 10
Number of records processed in one second	5/2000: 719 1/10,000: 884	25/2000: 843 5/10,000: 934	50/2000: 850 10/10,000: 885
Processing time on server (seconds)	5/2000: 2.23 1/10,000: 2.01	25/2000: 12.01 5/10,000: 10.05	50/2000: 23.35 10/10,000: 20.01
Memory used during processing (MB)	5/2000: 3.01 1/10,000: 2.82	25/2000: 111.23 5/10,000: 108.50	50/2000: 199.17 10/10,000: 196.02

Table 8. *Apex Queueable* processing results for *delete* operation.

	10,000 records: 5 turns by 2000 1 turn by 10,000	50,000 records: 25 turns by 2000 5 turns by 10,000	100,000 records: 50 turns by 2000 10 turns by 10,000
Total time of transaction duration (seconds)	5/2000: 13.90 1/10,000: 11.31	25/2000: 94.25 5/10,000: 87.50	50/2000: 238.50 10/10,000: 179.93
Number of SOQL queries	5/2000: 5 1/10,000: 1	25/2000: 25 5/10,000: 5	50/2000: 50 10/10,000: 10
Number of DML operations	5/2000: 5 1/10,000: 1	25/2000: 25 5/10,000: 5	50/2000: 50 10/10,000: 10
Number of records processed in one second	5/2000: 719 1/10,000: 884	25/2000: 530 5/10,000: 571	50/2000: 419 10/10,000: 556
Processing time on server (seconds)	5/2000: 2.23 1/10,000: 2.01	25/2000: 3.92 5/10,000: 3.81	50/2000: 7.17 10/10,000: 7.02
Memory used during processing (MB)	5/2000: 1.85 1/10,000: 1.82	25/2000: 31.5 5/10,000: 31.1	50/2000: 63.41 10/10,000: 63.90

The *delete* operation for the Apex Queueable method worked very quickly. It used the smallest processing memory from the ones studied so far cases. The processing time on the server was also the smallest (Table 8). However, as for the *insert* operation, the wait time of server resource allocation was not measured. Only the time of the transaction was taken into account. With a busy server, Apex Queueable processing may turn out to be less efficient.

Table 9. *Apex Queueable* processing results for *update* operation.

	10,000 records: 5 turns by 2000 1 turn by 10,000	50,000 records: 25 turns by 2000 5 turns by 10,000	100,000 records: 50 turns by 2000 10 turns by 10,000
Total time of transaction duration (seconds)	5/2000: 12.55 1/10,000: 11.61	25/2000: 62.75 5/10,000: 56.12	50/2000: 136.60 10/10,000: 118.91
Number of SOQL queries	5/2000: 5 1/10,000: 1	25/2000: 25 5/10,000: 5	50/2000: 50 10/10,000: 10
Number of DML operations	5/2000: 5 1/10,000: 1	25/2000: 25 5/10,000: 5	50/2000: 50 10/10000: 10
Number of records processed in one second	5/2000: 816 1/10,000: 862	25/2000: 796 5/10,000: 891	50/2000: 732 10/10,000: 841
Processing time on server (seconds)	5/2000: 2.56 1/10,000: 2.32	25/2000: 13.45 5/10,000: 11.05	50/2000: 27.54 10/10,000: 26.83
Memory used during processing (MB)	5/2000: 1.24 1/10,000: 1.11	25/2000: 62.50 5/10,000: 61.66	50/2000: 124.51 10/10,000: 123.21

Processing large amounts of data with Apex Queueable requires dependency on record transactions—Transaction Chain. Each processing queue must call the next when it is finished. When several independent processing queues are started at the same time, they compete for resources. Records are locked by the first queue to access them. When the next queue tries to modify the locked records at the same time, the system generates an error.

Thanks to the e-mail messages sent by the platform, the user does not have to monitor background processes all the time. In the event of errors, the precise information on their cause is provided to the person who started the process. Information about processing Apex Queueable can also found in the Salesforce Developer Console logs. Logs written to the console indicate exactly which class, method, and line of code caused the error. This allows the developer to quickly start working on fix the problem.

Like the *delete* operations, there is a very efficient *update* operation for an Apex Queueable method. It worked faster and used less memory than previously studied methods (Table 9). For each of the consecutive processing queues, the number of DML (data manipulation language) performed operations is counted from beginning. This allows one to bypass data processing limits imposed for the force.com platform.

5.4. Apex Schedulable Method

This method allows one to schedule the run a maximum of 100 tasks at the same time. Each Scheduled Job can process to 10,000 records. The collected data processing results are presented in Tables 10–12.

Table 10. *Apex Schedulable* processing results for *insert* operation.

	10,000 records: 5 turns by 2000 1 turn by 10,000	50,000 records: 25 turns by 2000 5 turns by 10,000	100,000 records: 50 turns by 2000 10 turns by 10,000
Total time of transaction duration (seconds)	5/2000: 25.62 1/10,000: 22.60	25/2000: 143.75 5/10,000: 135.45	50/2000: 291.89 10/10,000: 252.20
Number of SOQL queries	In both cases: 0	In both cases: 0	In both cases: 0
Number of DML operations	5/2000: 5 1/10,000: 1	25/2000: 25 5/10,000: 5	50/2000: 50 10/10,000: 10
Number of records processed in one second	5/2000: 390 1/10,000: 440	25/2000: 347 5/10,000: 369	50/2000: 342 10/10,000: 396
Processing time on server (seconds)	5/2000: 2.52 1/10,000: 2.41	25/2000: 16.55 5/10,000: 12.65	50/2000: 24.73 10/10,000: 20.81
Memory used during processing (MB)	5/2000: 1.92 1/10,000: 1.91	25/2000: 98.91 5/10,000: 95.54	50/2000: 192.34 10/10,000: 191.02

The *insert* operation for Apex Schedulable is slower than for Apex Batchable, Apex Queueable, or Future Apex. For large transactions (the processing of 50,000 and 100,000 records), memory usage increased (Table 10). The tasks converters were started at the same time. Every job must have a different name; therefore, it can be concluded that each of the tasks should be intended to perform another operation. Processing tasks (called Scheduled Jobs) are therefore not designed to perform operations with large amounts of data. Another proof of this is the fact that Apex Schedulable processing, just like Future Apex, causes record locking. If the user would like this method to process a large amount of data, it would have to run the jobs at intervals ensuring that the previously planned task has been completed. It is very inefficient. Unfortunately, it cannot be guaranteed that the processing ends by specific moment. Tasks cannot be added to the queue or made dependent on each other. Therefore, the results of this type of processing were only performed for 10,000 records. For the *update* and *delete* operations, an additional interval of 60 s was added. Thanks to this, there was no blocking of records (Tables 11 and 12).

Table 11. *Apex Schedulable* processing results for *delete* operation.

	10,000 Records: 5 Turns by 2000; 1 Turn by 10,000
Total time of transaction duration (seconds)	5/2000: 19.01 + 4*60 s of interval 1/10,000: 22.62
Number of SOQL queries	5/2000: 5; 1/10,000: 1
Number of DML operations	5/2000: 5; 1/10,000: 1
Number of records processed in 1 s	5/2000: 526; 1/10,000: 442
Processing time on server (seconds)	5/2000: 0.86; 1/10,000: 0.761
Memory used during processing (MB)	5/2000: 0.63; 1/10,000: 0.62

Table 12. *Apex Schedulable* processing results for *update* operation.

	10,000 Records: 5 Turns by 2000; 1 Turn by 10,000
Total time of transaction duration (seconds)	5/2000: 25.61 + 4*60 s of interval 1/10,000: 24.44
Number of SOQL queries	5/2000: 5; 1/10,000: 1
Number of DML operations	5/2000: 5; 1/10,000: 1
Number of records processed in 1s	5/2000: 390; 1/10,000: 409
Processing time on server (seconds)	5/2000: 2.66; 1/10,000: 2.41
Memory used during processing (MB)	5/2000: 1.25; 1/10,000: 1.23

Processing the *delete* and *update* operations is also less efficient than processing the same operations for previously tested methods. Additionally, an appropriate time interval had to be added, which lengthens the overall time even further during the transaction (Tables 11 and 12).

5.5. Comparison and Analysis of the Methods

Tables 13–15 show the largest collected numbers of processed records in one second for all the abovementioned methods.

For the *insert* operation, the best results were obtained during Apex Queueable processing (Table 13). The slowest data were processed using Future Apex methods. For the Apex Trigger method, it was not possible to collect data on processing 50,000 and 100,000 records.

Table 13. The number of records processed in one second for *insert* operation.

	10,000 Records	**50,000 Records**	**100,000 Records**
Apex Batchable	653	612	635
Future Apex	399	359	292
Apex Queueable	884	934	885
Apex Schedulable	440	369	396
Apex Trigger	420	-	-

For the *delete* operation, the best results were collected during Apex Queueable processing (Table 14). Unfortunately, most of the tested methods failed to collect results for processing 50,000 and 100,000 records. This is most commonly caused by mutual locking of records.

For the *update* operation, no results could be collected either for processing of 50,000 and 100,000 records for Future Apex, Apex Schedulable, and Apex Trigger methods. The best results of all tested processing methods were collected for Apex Queueable (Table 15).

Table 16 summarizes the longest and the shortest data processing time on the server for 10,000 and 100,000 records.

Table 14. The number of records processed in one second for *delete* operation.

	10,000 Records	**50,000 Records**	**100,000 Records**
Apex Batchable	412	423	434
Future Apex	455	-	-
Apex Queueable	884	571	556
Apex Schedulable	526	-	-
Apex Trigger	524	-	-

Table 15. The number of records processed in one second for *update* operation.

	10,000 Records	**50,000 Records**	**100,000 Records**
Apex Batchable	641	579	715
Future Apex	367	-	-
Apex Queueable	862	891	841
Apex Schedulable	409	-	-
Apex Trigger	422	-	-

Table 16. The shortest and the longest processing time (seconds) on the server for data processing methods.

10,000 Records	***Insert* Operation**	***Update* Operation**	***Delete* Operation**
The longest time	1/10,000: 3.02	1/10,000: 2.87	5/2000: 3.1
Processing method	*Apex Trigger*	*Future Apex*	*Future Apex*
The shortest time	5/2000: 1.9	1/10,000: 2.32	5/2000: 0.7
Processing method	*Apex Batchable*	*Apex Queueable*	*Apex Batchable*
10,000 records	***Insert* operation**	***Update* operation**	***Delete* operation**
The longest time	500/200: 35.61	50/2000: 27.54	500/200: 11.54
Processing method	*Apex Batchable*	*Apex Queueable*	*Apex Batchable*
The shortest time	10/10,000: 20.01	50/2000: 26.21	10/10,000: 7.02
Processing method	*Apex Queueable*	*Apex Batchable*	*Apex Queueable*

The best data processing results were collected for the Apex Queueable method and Apex Batchable method. For 100,000 records, only the two methods were compared, because for the others, it was not possible to collect the test results (Table 16). Taking into account the results collected during the research, the following conclusions were drawn:

Apex Batchable and Apex Queueable are best at processing of large amounts of data. By using them, we can bypass the limits of force.com data processing. The more transactions are running, the slower the data are processed. Each data processing operation requires additional operations server resources. Consequently, the more data are processed during one transaction, the faster the whole process is completed. This rule works well for all tested methods.

Even Apex Queueable is apparently processing the records faster, it is possible that with a busier server it works much slower. This may happen due to the fact that each queued transaction must wait for the completion of preceding transaction operations. Apex Batchable works slower than Apex Queueable; because of that, records are divided into groups of up to 2000 records. Apex Queueable can process records in groups of 10,000 records. Apex Batchable has no record locking issues. This is because the records to be processed are retrieved from the database only once in the *start()* method of the Batchable interface. Later, these records remain divided into groups and separated into the correct number of processes. These processes are independent of each other. Additionally, Apex Batchable always performs one SOQL (Salesforce object query language) operation less than the other processes. What happens in the *start()* method does not count toward the limits transaction processing. Apex Batchable is suitable for doing the same operations on a large number of records and guarantees a quick completion of the whole processing.

Apex Queueable can be useful not only when processing a large group data but also in the processing of transactions that are dependent on each other—when the start of the second transaction depends on whether the previous one was successful. Each operation requires the operation to be completed predecessor, which is guaranteed by the use of Apex processing Queueable along with making queued tasks dependent on each other (Transaction Chain).

Apex Batchable, Apex Queueable, Scheduled Apex, and Future Apex represent asynchronous processing. They run in the background not preventing system users from performing other duties. This type of processing is suitable for long, complex performance operations—the result of which the user may find out later time. These can be operations such as making monthly summaries or generating reports and documents. Apex Schedulable additionally allows developers to set an exact date when the processing takes place to begin. It is also possible to set the repeatability of a given process.

Triggers are not suitable for processing a large number of records. Process they data synchronously, therefore they are treated as a single data transaction. Data Manipulation (DML) operation cannot be performed Language) with more than 10,000 records. Triggers are suitable for tasks such as small data processing and quick return responses to the user. When using the application, users modify forms, save changes to individual records, and delete and add new data. By using triggers, developers can control what happens before or after users execute DML operations.

Future Apex processes data asynchronously but is not suitable for performing the same operation on a large number of records. Running the operation is not addictive; therefore, there is no way around the problem of mutual blocking of transactions. Future Apex can stay used, for example, when implementing the integration of two systems. The code in the method with the *@future* annotation is executed when the given process allocates processor resources. This method may attempt to communicate with another system, wait for its response, and save it to base. At this time, the rest of the code of the currently processed process is not blocked, and the user can continue to work without waiting for response from an external system.

Each of the tested processing methods has its advantages and disadvantages. It is important to understand well how they work. This allows them to be the appropriate use for the benefit of the enterprise. The incorrect use of methods of data manipulation can contribute to the occurrence of various problems such as delays, irregularities in reports, and the need to introduce costly changes in system implementation.

6. Discussion and Conclusions

The methods of data processing and storage described in the paper support the work of enterprises every day. The choice of how to process and store information is an individual matter and depends largely on the size and type of business of the company. The technological advances in the development of servers, memory, and networks are changing not only the infrastructure of data centers but also the concepts of its architecture. Data centers are undergoing a constant transformation, and the data centers that will begin to emerge in the near future will look completely different from what we know today. This is mainly due to the development directions of companies that must constantly change in order to meet market requirements. Expectations are constantly growing so that IT systems can be flexibly adapted to the current needs and dynamically scaled. One of the most important factors influencing the evolution of data centers is the development of relatively new technologies that are quickly gaining popularity or will be used in mass in the near future.

Cloud computing supports the work of the companies around the world. Due to its numerous advantages, its popularity continues to grow. Companies that start to invest in this type of service must carefully study the variety offers of service providers. The range of possibilities that can be used is huge; therefore, it is important to think carefully about what it really is that the company needs and how much data it would like to store and process with clouds. Data security is also very important. Please note that storing information in your own data center does not always guarantee greater protection. Sometimes, an experienced service provider who has dealt with for years securing their clients' data would be able to provide and implement more security mechanisms than the company itself is trying to protect its data from attacks by cyber criminals.

The service provider in the Salesforce.com cloud model offers many ways to manipulate data on your platform. In an application created under of this thesis, the results of the insert, update, and operation were collected and collected to delete for the five selected processing methods. After analyzing the results, it was specified for what purposes companies can use these methods:

Processing large amounts of data—the best method for processing large amounts for the number of records is Apex Batchable and Apex Queueable. Both of these methods they can process huge amounts of data without blocking work platform users or other business processes. Apex Queueable, additionally, allows for the dependence of several processes on each other. Every operation will wait for the end of the previous one before starting. Thankfully, this company can be sure that the end result of the processing will be complete and that nothing will be missed.

Performing recurring modifications—Apex Schedulable is best suited or modifications that are constantly performed at given intervals. This method allows one to choose at what time and for how much time the process is run. The platform allows to run many such tasks at the same time. The only thing to remember is that processes do not modify the same records at the same time. The operation in such case will be aborted, and an exception will be generated.

Data processing through integration with external systems—Future Apex methods works well for communication with external system. This processing method runs the process in separate thread so as not to block the user's work. This is useful, seeing as waiting for a response from another system may often take a while. An additional advantage of this way of processing is reduced platform limits on database query execution and memory usage. Sequential data processing—if the user needs to process data in one thread, so that all instructions are executed sequentially, the user should use trigger processing. For the user who will start the process, the user will not be able to perform any further operations until response from the first process will not be returned. Accordingly, this type operations should not process large groups of records.

Processing a large number of tasks with the ability to monitor them—all asynchronous processes (Apex Batchable, Apex Queueable, Future Apex, and Apex Schedulable) can be monitored using the force.com platform. System user can observe the course of running processing, view its operation status, and read out any errors. Additionally, the information about problems can be read using the development console, which allows one to view system logs.

Data processing methods can be combined. This allows one to take advantage of different solutions at the same time. An example of such a connection can be the use of Apex Schedulable together with Apex Batchable. The company can need to process a large number of records that must be performed periodically. Using Apex Schedulable, the start date and periodicity of the process can be set exactly. The Apex Schedulable process may run an *execute()* method of a class that implements the Batchable interface. This will make it possible to process a large number of records.

Author Contributions: Conceptualization, M.S. and A.P.-M.; methodology, M.S. and A.P.-M.; validation, A.P.-M. and W.M.; formal analysis, M.S. and A.P.-M.; investigation, M.S.; resources, M.S.; data curation, M.S.; writing—original draft preparation, A.P.-M. and W.M.; writing—review and editing, A.P.-M. and W.M.; visualization, A.P.-M. and W.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Poniszewska-Maranda, A.; Matusiak, R.; Kryvinska, N.; Yasar, A. A Real-time Service System in the Cloud. *J. Ambient Intell. Human. Compu.* **2020**, *11*, 961–977. [CrossRef]
2. Carey, C.; Raisinghani, M.S.; White, B. Foundations of Data Center: Key Concepts and Taxonomies. In *Engineering and Management of Data Centers. Service Science: Research and Innovations in the Service Economy*; Marx Gómez, J., Mora, M., Raisinghani, M., Nebel, W., O'Connor, R., Eds.; Springer: Cham, Switzerland, 2017.
3. Chang, W.Y.; Abu-Amara, H.; Sanford, J.F. Cloud Service Business Scenarios and Market Analysis. In *Transforming Enterprise Cloud Services*; Springer: Berlin/Heidelberg, Germany, 2011.
4. Data Processing and Data Processing Methods. Available online: https://planningtank.com/computerapplications/data-processing-data-processing-method (accessed on 1 September 2020).
5. What Are the Steps in Business Data Processing. Available online: https://www.dataentryoutsourced.com/blog/what-are-the-steps-in-business-dataprocessing (accessed on 1 October 2020).
6. Basic Bisuness Data Processing Principles. Available online: https://smallbusiness.chron.com/basicbusiness-data-processing-principles-21983.html (accessed on 1 October 2020).
7. Lalitha, V.P.; Sagar, M.Y.; Sharanappa, S.; Hanji, S.; Swarup, R. Data security in cloud. In Proceedings of the International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 1–2 August 2017; pp. 3604–3608. [CrossRef]
8. Alani, M.M. General Cloud Security Recommendations. In *Elements of Cloud Computing Security*; SpringerBriefs in Computer Science Series; Springer: Cham, Switzerland, 2016.
9. Balobaid, A.; Debnath, D. Cloud Migration Tools: Overview and Comparison. In *Services—SERVICES 2018*; Yang, A., Kantamneni, S., Li, Y., Dico, A., Chen, X., Subramanyan, R., Zhang, L., Eds.; LNCS Volume 10975; Springer: Berlin/Heidelberg, Germany, 2018.
10. Sehgal, N.K.; Bhatt, P.C.P. *Cloud Computing. Concepts and Practices*; Springer: Cham, Switzerland, 2018.
11. Seruga, J.; Hwang, H.J. Cloud Computing for Business. In *Software and Network Engineering*; Studies in Computational Intelligence; Lee, R., Ed.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 413.
12. Importance of Data Processing. Available online: https://planningtank.com/computer-applications/importance-of-data-processing (accessed on 1 September 2020).
13. Cloud vs. Data Center. Available online: https://www.businessnewsdaily.com/4982-cloud-vs-datacenter.html (accessed on 1 November 2020).
14. Types of Data Processing. Available online: https://planningtank.com/computer-applications/types-ofdata-Processing (accessed on 2 January 2021).
15. Mohamed, B.; Mostapha, Z.; Mohamed, E.; Pierre, M. Cloud Computing, IoT, and Big Data: Technologies and Applications. In *Concurrency and Computation: Practice and Experience; Special Issue: Special Issue on Computational Intelligence Techniques for Industrial and Medical Applications (CITIMA2018), Las Palmas de Gran Canaria, Spain, 26–29 November 2018*; Wiely: Hoboken, NJ, USA, 2020; Volume 32.
16. Okkyung, C.; Wooyeol, J.; Kangseok, K.; Yeh, H. Mobile cloud computing model for data processing and analysis on smartphone. In Proceedings of the 2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012), Taipei, Taiwan, 23–25 October 2012; pp. 116–119
17. Manoj, M.; Tilak, A.; Charles, T. Big data in Cloud computing review and Opportunities. *Int. J. Comput. Sci. Inform. Technol. IJCSIT* **2019**, *11*, 4.
18. Deepak, P.; Sahoo, B.; Mishra, S.; Swain, S. Cloud Computing Features, Issues, and Challenges: A Big Picture. In Proceedings of the 2015 International Conference on Computational Intelligence and Networks, Odisha, India, 12–13 January 2015; pp. 116–123.
19. De Donno, M.; Koen, T.; Dragoni, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog. *IEEE Access* **2019**, *7*, 150936–150948. [CrossRef]
20. Ameen, A.; Liu, H.; Zuo, D.; Basheer, A. Cloud Computing: A model Construct of Real-Time Monitoring for Big Dataset Analytics Using Apache Spark. *J. Phys. Conf. Ser.* **2017**, *933*, 012018;
21. Forestiero, A.; Mastroianni, C.; Meo, M.; Papuzzo, G.; Sheikhalishahi, M. Hierarchical Approach for Green Workload Management in Distributed Data Centers. IEEE Trans. Green Commun. Netw. **2017**, *1*, 97–111. [CrossRef]
22. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. *Mobile Edge Computing—A Key Technology towards 5G*; ETSI White Paper; European Telecommunications Standards Institute: Valbonne, France, 2015; Volume 11, pp. 1–16.
23. Lee, J.; Kim, J.-W.; Lee, J. Mobile Personal Multi-Access Edge Computing Architecture Composed of Individual User Devices. *Appl. Sci.* **2020**, *10*, 4643. [CrossRef]
24. Wang, T.; Mei, Y.; Liu, X.; Wang, J.; Dai, H.N.; Wang, Z. Edge-based auditing method for data security in resource-constrained internet of things. *J. Syst. Arch.* **2021**, *114*, 101971. [CrossRef]
25. Lee, J.; Kim, D.; Lee, J. Zone-based multi-access edge computing scheme for user device mobility management. *Appl. Sci.* **2019**, *9*, 2308. [CrossRef]
26. Apex Documentation. Available online: https://developer.salesforce.com/docs/atlas.enus.apexcode.meta/apexcode/apex_intro.htm (accessed on 2 January 2021).