Improvement of Parallelism Process in Distributed Data Processing

Roman, Ceresnak
University of Zilina
Zilina, Slovakia
roman.ceresnak@fri.uniza.sk

Karol, Matiasko
University of Zilina
Zilina, Slovakia
karol.matiasko@fri.uniza.sk

***Abstract*—Many sectors have been related to massive population growth, whether healthcare, industry, transport, or information systems. Many of these industries daily generate a vast amount of data, and a basic system of the data processing stopped to fulfill efficiencies of data manipulation. The central data processing also has various disadvantages, such as central storing of the server, low storability, and high costs. Because of these, but also another reasons, the system called distributed data processing was created. Distributed data processing led to calculations' acceleration, higher redundancy, and bigger storability. The massive use of the calculation servers happens during distributed data processing with big data, causing unsatisfactory performance. Based on observations, we created an architecture capable of effectively adding and deleting the calculation units based on demand in the data processing and increasing resistance against an error. The experiments show that the newly created architecture can increase security, not only of the data processing but also of processed data security. On the other hand, adaptation to the performance plays a critical role we achieved after performing the planned experiments.**

I. Introduction

Distributed data processing is a method of computer network, where more computers in different places, share options for computer processing. This is the difference opposite the only one centralized server managing and providing the options of the processing to every connected system. The computers forming the distributed network for data processing are placed in different places, but they are connected through wireless or satellite connections.

The data exploding in last years result in increasing demand after the big data processing in modern data centers, usually distributed in different geographic regions, for example in the 13 data centers of Google company in the 8 countries on 4 continents [1]. The big data analysis proved its significant potential to uncover valuable knowledge about the data, improve decision-making, minimalize risks, and develop new projects and services. On the other hand, the big data were already translated into the high price, because of high demand after the calculation and communication sources [2]. The researchers assume, the costs on hardware data centers will still grow, which is related to its high costs of efficiency of increasing and customer demand. This is why it is necessary to study the problem of cost minimalization for the big data processing distributed in many data centers.

Several methods were developed to optimize the prices, regarding optimal calculation units. It was suggested, for example, to change the data center size to reduce the calculation costs by the edit of the number of activated servers through task placement [3]. Based on the data center change, the study was created, focused on the geographic deployment of the data center [4], [5]. The problem of distributed processing data is related also to its effectively created communication. We noticed the studies devoted to the mentioned problem to improve the data through the task placement in the server, where the input data are placed to prevent distant data loading [6].

Even if the studies, as mentioned above, brought specific positives in distributed data processes, they are diametrically different from the system with cost-effective data processing because of the following reasons:

- Source waste – to details, which occurs in the system, is related to appropriate treatment. Some data are occurring in the system more often, some data are accessed less often, so it is unnecessary to have them constantly available. This fact helps us to effectively manipulate the data, which are no often demanded; thus, it will reduce the costs, or it will reduce the number of the servers needed for processing,
- Connection in networks – the data center's velocity is influenced by several factors related to their cost characteristic [7]. However, the existing strategy of the direction between the data centers does not use the variety of network connections in the data centers. This is because, because of a limitation of storing capacity and calculations, storing all tasks to the same server is not possible, where relevant data are stored. It is necessary to withdraw certain data from the distant server,
- The data's inefficiency – in the system, there are the data, which replication can also depend on its appropriate use. Often used data demands bigger replication because of their key role than the data inappropriate to access, thus the reduction of individual servers overload could happen. Also, an amount of consumed storage could be reduced.

We study the cost problem for big data processing to overcome these weaknesses of the existing solutions and the efficiency of the data replication between several servers. We

take the limitation of the provided number of server into account in our cost optimization. We aim to optimize a vast amount of the data to eliminate the needed replicas amount. Our primary papers are summarized like this.

- Optimize the number of parallel processes used in processing the significant data amount, thus preventing excessive server overload.
- Optimize the replicated data amount, based on individual data weight, thus optimizing the number of servers needed to manage the data.

II. Related work

Optimization of parallelism

Nowadays, it is possible to use many developers such as Hadoop, Spark, Flink, Samza, and many others, when using distributed data processing. A data division is used in these systems to manage parallelism, and it is central for these systems to achieve the scalability of significant calculation clusters. However, the systems' diversion techniques are very primitive, which causes severe problems regarding the performance. In the paper focused on the given problematic, researcher Schneider introduced a compilator and runtime system, together with the team [8], which automatically extracts the data parallelism in distributed flow processing. According to the available resources, their access to a compilation of the parallel areas, the compilator ensures security. Thereby it takes an operator selectivity, state, diversion, and dependence on the operators into account. A distributed runtime system ensures that n-tuples always leave the parallel areas in the same order they could be without the data parallelism by using the most effective strategy identified by the compilator. Another point of view on the given problem has researcher Dean together with Ghemawat [9], who came up with an idea of using MapReduce implementation running in a big group of commodity machines. They transferred the calculation to various devices, and they use cluster by Google in the research.

Price optimization

Until recently, the vast data centers were used to calculate the massive amount of the data. The data centers provide various calculation purposes in many ways, which is related to their cost, too. According to the [10], the data center can be composed of many servers and can consume megawatts of energy. Electricity cost price is a big negative in processing the vast data amount for the data center providers. This is why significant attention of the academic community and the industry [11], [12], focused on reducing electricity costs. Between the mechanisms developed until recently for energy management in the data centers, the techniques attracting significant attention are task placement and DCR.

DCR and task placement are usually considered as typical to meet the requirements of calculations. Liu and col [13] repeatedly review the same problem, including delays in the network, Fan et al. [11] examine the strategies of providing energy, how much of a calculation device is possible to safely and effectively place, regarding given budget and energy. Rao and col. Examine how to reduce the electricity costs by the user's demands movement to geographically distributed data centers with relevantly actualized sizes meeting the demands. A very different idea for the price optimization was given by researchers like Sharon Q. [14] and the collective focusing and comparing the efficiency of the transfer from a physical server to a cloud service. The mentioned study compares centralized data centers, virtual servers, and safe data transfer through the internet. Another researcher from the scientific community Qian addressed a similar problematic together with the collective [15], who published the paper focused on the definition of the term, history, advantages, and disadvantages of cloud computing and the definition of the value chain and efforts on standardization.

Data protection

Nowadays, data protection belongs to an essential part of the development of whichever system. We noticed four key works during our research from researchers Agarwal, Cidon, Schachnai, and Jin together with their collectives, addressing given problematic. Agwal, together with his research team (A.P.Aakash 2020), designed a mechanism of the data placement, Volley for geographically distributed cloud services. This work takes costs on bandwidth WAN, interdependencies, limitation of the data center capacity, etc. into account. Volley analyses protocols based on the so-called "interactive optimizing algorithm." The algorithm is based on the access data and client placements. The mentioned paper also provides the migration recommendations back to the cloud service. The second exciting study from researchers Cidon and his collective [16] is related to a mechanism named by MinCopysets. The algorithm is based on the effective placement of the data replicas because of making the placement of data durability more effective in distributed data centers. The third mentioned by the researcher and the collective [17] examines how it is possible to minimalize the costs related to communication while ensuring the comfort by the end-user, based on various video files' copies placement on servers and with that relating loading capacity of the associated copy. The recently published study by Jin, together with the collective [18], designed a standard scheme optimization, optimizing virtual machine placement and providing recommendations with the direction of network flow because of energy saving.

Reliability

In studying the system's reliability, we also examined the problem dealing with the sensitivity and accuracy of the problem provided. In the paper [20], the author deals with managing the temporary system's granularity and proposes a data-sharing model based on the reliability, sensitivity, and accuracy of data providers. It provides a system concept that introduces a cash prospect, which is then evaluated in the experiment section. Optimization of the data flow by historical data aggregation and limitation of the data amount is a core part for the system decision making, whereas the time for data transferring is strictly limited.

Another look at increasing reliability is given in the studies in the article [21]. The main idea is to manage the data asynchronously, and then the data is merged. Study data began in 2006 and is the result of an in-depth analysis. The study's achieved result is the creation of architectural design for a distributed information system with asynchronous update data. During the development of the researchers concluded came the need to store versioned data on the server. Their different approach to solving reliability in the mentioned paper uses new techniques of storing versioned data in a unitemporal relational

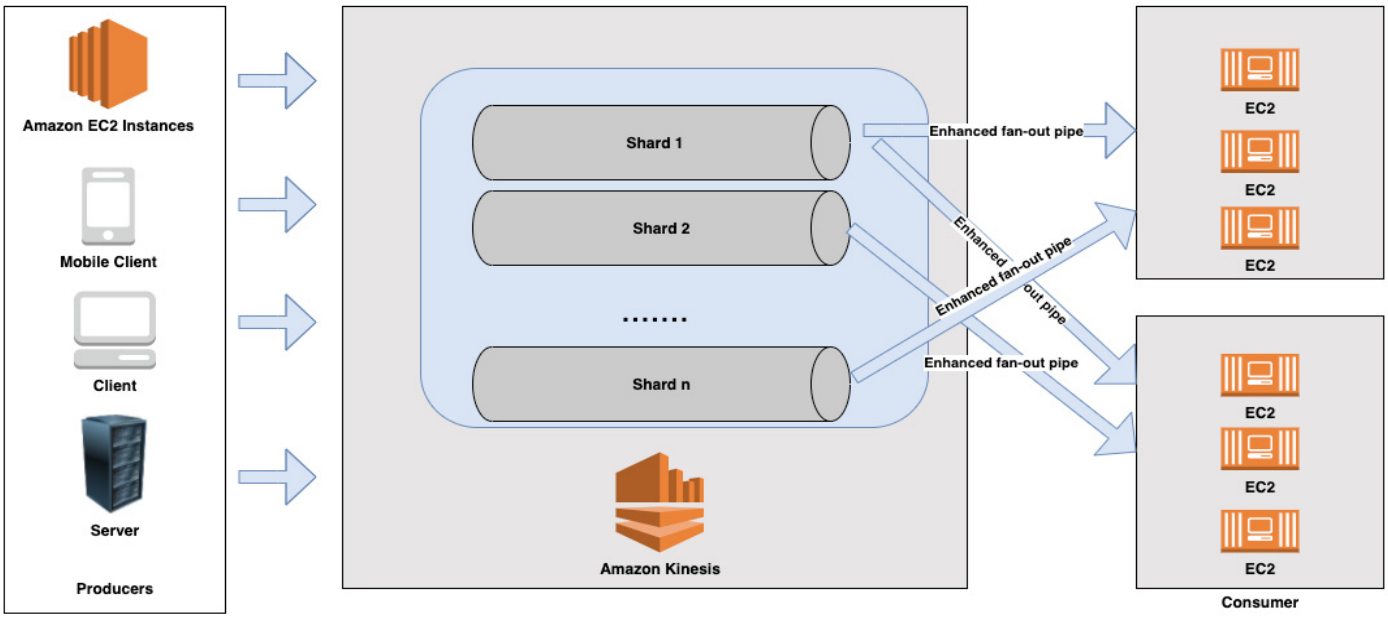

Fig 1. Kinesis Architecture

database. Storage is a departure from traditional security practices. The created solution can also preserve the advantages of RDBMS, such as referential integrity and transaction processing.

The rest of the paper is structured as follows. Related works are summarized in the II part. Part III represents our system model. The cost optimization is formulated as problem MNLP in section IV and then, it is linearized in section V. Theoretic findings are proved by the experiments in section VI. Part VII competes with our work in the end.

III. Contribution

We used Amazon Kinesis Stream in cloud service Amazon for distributed data processing. Amazon Kinesis Streams is a permanent and scaling service in real-time.

- The consumers get records from Kinesis Data Streams and process them. We build our applications using either Kinesis Data Analytics, Kinesis API, or Kinesis Client Library (KCL).

For processing data in the Kinesis service, we created a script, which we saved at the following address https://github.com/romanceresnak/kinesis/blob/master/script.py .The mentioned script performs the following steps:

The values entering the system are portrayed in Fig 1 on the left. It is seen, the records are coming from different calculation units EC2, respectively, from any mobile application having API client access with the help of service Cognito.

The data processing velocity in the process portrayed in fig 1 depends on the efficiency of operation Kinesis. Because of this, we provided the flexibility of scaling to service Kinesis, based on overloading.

It can group gigabytes of the data in a second from hundreds of thousands of resources, including flows of the database events, flows of web clicks, financial transactions, protocols IT, social media channels, and location watching events. The caught data are given in milliseconds for the analytic data in real-time, including anomaly detections in real-time, dashboards in real-time, and dynamic price making. Of course, it is also the connecting to the service Kinesis with different computers and servers. Subsequently, the processing runs as follows:

- The producers put records (data ingestion) into KDS. AWS provides Kinesis Producer Library (KPL) to simplify producer application development and achieve high write throughput to Kinesis data stream.
- A Kinesis dataStream is a set of shards. Each shard has a flow of data records. Data records are formed of a sequence number, a partition key, and a data blob (up to 1 MB), an immutable bytes sequence

We created a script for the data processing in service Kinesis, which we uploaded to the following address https://github.com/romanceresnak/kinesis/blob/master/ script.py The mentioned script performs the following steps:

- At first, it will start up a timer to catch the time of the script performing.
- We create a client with Kinesis in the French region (eu-west-1), returns a client.
- Loads data in form the CSV renamed to ‘data.csv,’ returns a pandas *DataFrame*
- Each record's fields are combined utilizing a '|' (pipe) character, which we influence later. We cluster up all the information and post it Kinesis to be handled utilizing the customer we made, giving the Stream name and shard tally. Shards are turned through to take into consideration appropriate fanning out of the remaining task at hand. Notwithstanding, we are just utilizing one here.
- Subsequently, we stored the data into 2 data regions.

We stored the data into two regions, which means a replicative coefficient is set to value 2. We set an automatic data replication in our structure in the case of region loss. The number of copies is automatically found out in the case of the

region loss. If data replication is not equal to value 2, the data are automatically replicated to another region. Regarding the mentioned case, we set an automatic message IP address, which has the purpose of the region loss camouflage to avoid whichever data loss is noticed by the end-user

A. Scaling up policy

The automatic storing upwards allows the amount of the needed threads to adapt, based on the data increasing, that is to say traffic.

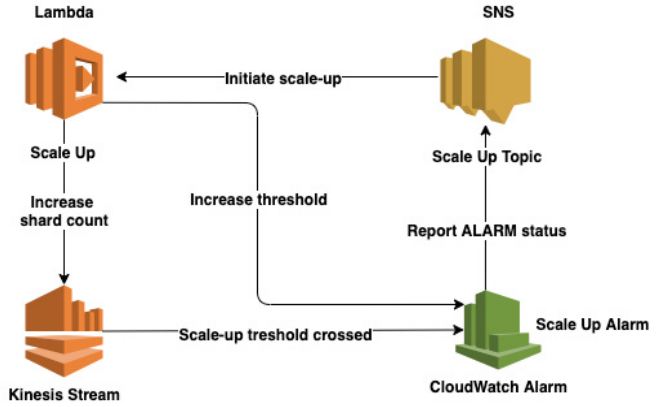

Fig 2. Scale Up Architecture

We created an architecture for scaling of thread amount portrayed in Fig 2. It comprises four services: Kinesis Stream, Lambda function, Simple Notified Service, and Amazon CloudWatch.

The alarm of service CloudWatch monitors metrics of Kinesis Data Stream service. When the alarm threshold is achieved, for example, because of demand growth, respectively, the number of threads, so some of the mentioned events will trigger the alarm. This triggering of the alarm will alert the automatic application of adaptation policy, which reacts by automatic scaling downwards, based on stated preferences.

At the point when the scaling strategy is set off, Application Auto Scaling calls an API activity. The consider passes the new number of Kinesis Data Stream shards for the ideal limit. The call additionally passes the name of the asset to scale, given by Amazon API Gateway. Amazon API Gateway summons an AWS Lambda work. Given the data sent by Application Auto Scaling, the Lambda work increments or diminishes the quantity of shards in the Kinesis Data Stream. It does so by utilizing Kinesis Data Stream's *UpdateShardCount* API activity. The accompanying chart represents the situation.

The Lambda will report two custom metrics (*OpenShards* and *ConcurrencyLimit*) to CloudWatch whenever strongly invoked to enable tracking when scaling up occurs. These usage metrics will allow us to monitor scaling behavior.

As recently referenced, the Scale-Up Lambda will utilize an alert to screen a Kinesis metric to check whether it crosses a determined limit.

The prescribed methodology is to quantify the whole of *IncomingRecords* or *IncomingBytes* from the related Kinesis stream for more than 5 minutes. This will give us direct knowledge into how much information streams into the stream and settles on educated choices concerning scaling.

B. Scaling down policy

A Lambda scales down the Kinesis stream, the scale-up alert, and alternatively an outside Lambda to their unique settings.

When daily during an off-top hour (after bombed logs have been prepared), a CloudWatch Rule will trigger the Scale Down Lambda at 10-minute stretches. This is done to balance the impediment Kinesis has for downsizing (the most reduced legitimate objective shard check is half of the current free shard tally). The whole architecture is shown in Fig 3.

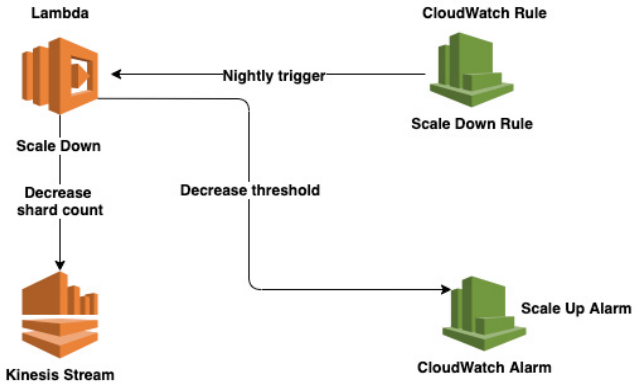

Fig 3. Scale down architecture

This Lambda will avoid the scale-down cycle if the stream is at present under heavy use, on the off possibility that it is as of now being downsized or on the off chance that it has just been downsized to the default number of shards.

Like the Scale-Up Lambda, the Lambda will likewise report two custom measurements (OpenShards and ConcurrencyLimit) to CloudWatch at whatever point is effectively summoned.

C. Threshold Calculation

In view of the requirement for the application, a programmed valuation of limit esteem occurs.

For Kinesis stream with n shards, Lambda will scale to all things considered n summons (as constrained by its held simultaneous executions).

Every Lambda sends a normal of m records to the Kinesis stream every second.

The time frame, by which the alert screens the whole of a measurement, is s seconds.

In this manner, the edge to screen is

$$n * m * s \quad (1)$$

To guarantee scaling up happens before the information falls behind, we can rather screen a level of the determined limit. Since 80% is viewed as best practice by AWS, we will screen that esteem as opposed to going ahead.

D. Cross-Region Replication

The values, which are processed, are to be stored after the successful manipulation effectively. The vast data amount is not possible to store on a central computer, and so they are stored distributed. The data loss could cause a protracted or alternative data loss because we work with a huge data amount.

The data in service Amazon are stored to Amazon S3, which reliability is defined as 99,99999999%. Even if the number is this high, it needs to preserve the reliability and data accessibility. That is why we decided to replicate the data between individual regions in the network.

We defined the replicative coefficient to value two by this method, which will automatically evoke the operations, in the case of error or failure, controlling the amount of replicated data in the system. If the replicative data coefficient is not equal to 2 after the region or any value failure, so the data are automatically replicated to other regions.

Based on the replicated data in two regions, we fulfilled the critical condition of preserving the critical data copies in places distanced hundreds of kilometers from each other. Our solution fulfills maintaining the strict regular demands to preserve sensitive financial and personal data, too.

E. Cross-Region Replication

We created 3 files with different file sizes and different data volumes for the purposes of this experiment. The files are uploaded on these addresses:

- https://github.com/romanceresnak/kinesis/blob/master/data2010-1000.csv
- https://github.com/romanceresnak/kinesis/blob/master/data2010-50000.csv
- https://github.com/romanceresnak/kinesis/blob/master/data2010-100000.csv

The created files are about the size of 1000, 50000, and 100000 records. All operations were done in the following configuration:

TABLE I. SERVER CONFIGURATION

EC2 Instance	a1.medium vCPU: 1 MeM(GiB): 2
EMR cluster	master: 1x m3.xlarge core: 2x m4.4xlarge

The values, we measured during the records processing about size 1000, 50 000 and 100 000, are as follows:

For *1000:*
Total Records sent to Kinesis: 1000
Runtime: 6.782348799345345 [s]

For *50000:*
Total Records sent to Kinesis: 50000
Runtime: 8.23432452345244 [s]

For *100000:*
Total Records sent to Kinesis: 100000
Runtime: 8.96464356323443 [s]

As seen in the results for the number of records 1000, 5000, and 100 000, the values did not degrade, and they did not grow exponentially with the increasing number of records. This fact is right influenced by the automatic adaptation of the calculation units based on an increasing number of data related to its increasing demand after the calculation units.

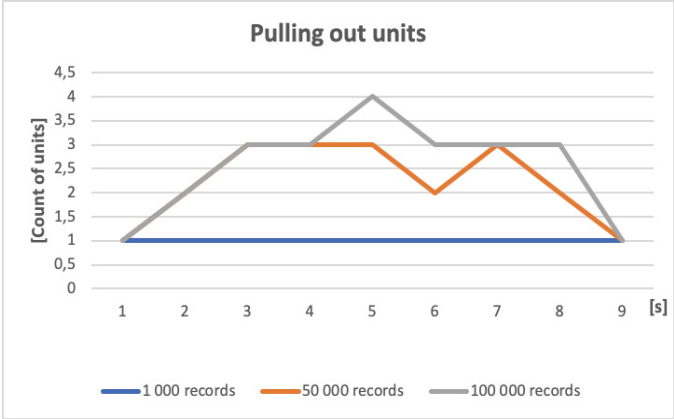

Fig 4. Pull out server drives

As seen in fig 4, the values we measured the size of 1000 records show that automatic scaling was not needed for these purposes because of the low data capacity. Our set value of the server overload was set at 80%, which was not exceeded in this value, and so any growth of the number of calculation units did not happen. Except for the mentioned fact, value 1 is always the value representing a minimal number of the calculation units, which are figured by the server at the beginning and the end of the process.

While comparing the values, which can be seen in fig 4, the number of calculation units increased by the data growth to 50000 records. As was mentioned above in the paper, the calculation unit occupancy crossed the border of 80% in the processing of 50000 values. Thus, it caused the growth of the calculation units. Even the number of calculation units that equals 2 was not enough to reduce the server occupancy, so another calculation unit was added again. With the number equaled to 3, the server occupancy reached the level when it was unnecessary to add or delete units. The server with optimal occupancy was able to process incoming data. The opposite effect happens after the processing of 2/3 records. The number of the calculation units needed to process the data started to decrease until it was not necessary to give only value 1 for the data processing. Therefore, the server overload did not optimize, and the cost optimization connected to the operation of the data processing.

We recorded a similar effect with the data growth to 100000 records, too. We reached the border equaled to 4 with the calculation units in automatic scaling after approximately half of the data. Subsequently, the data overload was decreasing, which is also related to the decreasing value of the calculation units towards 1.

As it is seen, the automatic adaptation of the performance works effectively either with several 1000 records or 100000. Thus, it helps us ensure a sufficient amount of the data to make the system work in optimal server overload with the inquiry on effective manipulation with operating costs.

IV. Conclusion

The majority of the accesses focus on the vast data amount in distributed data processing, which are in the system and create a new way to access the data effectively. Many researchers' effort is also to reduce the replicated data amount, thus reducing the hardware amount needed for the value storing to the system. This document has another attitude and shows how to effectively manipulate the data in the system and the data entering the system in real time. Based on achieved experiments related to the data replication, we do not share the same opinion with other researchers claiming the replicative coefficient has to be set minimally to value 3. Based on the experiments we assume, the replication coefficient must be set to value 2, which leads to data storage reduction, decreasing the calculation units, and of course to the cost optimization. The critical part is the data processing on devices geographically and physically separable and is connected by a highly available connection. This aspect makes our access unique in comparison with for example the work [19].

According to our experiences, automatic adaption lasts upwards only a few seconds, causing an increasing amount of operating costs. The solution was implemented in the paper, which adapts the calculation units number also downwards with decreasing calculation demand and because of effective use of the provided resources.

Automatic permission of the data replication is related to their appropriate management. The data watching with the replicative coefficient set to value 2 always signalizes that the system automatically starts to replicate the data in case of whichever data outage. The significant advantage is the setting of an elastic IP address, capable to mask the outage of the relevant area. Thus the user will not find that any outage happened.

Our designed architecture could be appropriate while using the applications, which do not know the data amount entering the process and needing the massive level of the performance variability while designing the application. We can imagine that entering data could automatically perform the structure's edit and influence the performance based on the structure type. We will try to apply these designs to our next work.

Acknowledgment

This work was supported by Grant System of University of Zilina No. 1/2020. (8056).

References

[1] J. Kamal, M. Murshed, and R. Buyya, “Workload-aware incremental repartitioning of shared-nothing distributed databases for scalable OLTP applications,” Futur. Gener. Comput. Syst., 2016.

[2] Y. Zhao, R. N. Calheiros, J. Bailey, and R. Sinnott, “SLA-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments,” in Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016, 2016.

[3] Q. Xia, W. Liang, and Z. Xu, “Data Locality-Aware Query Evaluation for Big Data Analytics in Distributed Clouds,” in Proceedings - 2014 2nd International Conference on Advanced Cloud and Big Data, CBD 2014, 2015.

[4] L. Zhang, T. Han, and N. Ansari, “Revenue driven virtual machine management in green datacenter networks towards big data,” in 2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings, 2016.

[5] E. Le Merrer and N. Le Scouarnec, “Efficient user opt-out from block stores,” in Proceedings - 2016 IEEE International Conference on Cloud Engineering Workshops, IC2EW 2016, 2016.

[6] L. Gu, D. Zeng, S. Guo, and B. Ye, “Joint optimization of VM placement and request distribution for electricity cost cut in geo-distributed data centers,” in 2015 International Conference on Computing, Networking and Communications, ICNC 2015, 2015.

[7] I. Marshall, “Linking cache performance to user behaviour,” Comput. Networks, 1998.

[8] S. Schneider, M. Hirzel, B. G. Gedik, and K. L. Wu, “Auto-parallelizing stateful distributed streaming applications,” in Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT, 2012.

[9] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in OSDI 2004 - 6th Symposium on Operating Systems Design and Implementation, 2004.

[10] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, “Cutting the electric bill for internet-scale systems,” in Computer Communication Review, 2009.

[11] X. Fan, W. D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” in Proceedings - International Symposium on Computer Architecture, 2007.

[12] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, “Benefits and limitations of tapping into stored energy for datacenters,” in Proceedings - International Symposium on Computer Architecture, 2011.

[13] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, “Greening geographical load balancing,” IEEE/ACM Trans. Netw., 2015.

[14] S. Q. Yang, “Move into the Cloud, shall we?,” Library Hi Tech News. 2012.

[15] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud computing: An overview,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009.

[16] A. Cidon, R. Stutsman, S. Rumble, S. Katti, J. Ousterhout, and M. Rosenblum, “MinCopysets : Derandomizing Replication In Cloud Storage,” Networked Syst. Des. Impementation, 2013.

[17] X. Zeng et al., “SLA Management for Big Data Analytical Applications in Clouds,” ACM Comput. Surv., 2020.

[18] H. Jin et al., “Joint host-network optimization for energy-efficient data center networking,” in Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013, 2013.

[19] M. Scavuzzo, E. Di Nitto, and S. Ceri, “Interoperable data migration between NoSQL columnar databases,” in Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW, 2014.

[20] M. Kvet, “Data Distribution in Ad-hoc Transport Network,” presented at the 2019 International Conference on Information and Digital Technologies (IDT), Jun. 2019, doi: 10.1109/dt.2019.8813437.

[21] J. Janech, M. Tavac, and M. Kvet, “Versioned database storage using unitemporal relational database,” presented at the 2019 IEEE 15th International Scientific Conference on Informatics, Nov. 2019, doi: 10.1109/informatics47936.2019.9119269