Optimal Geospatial Query Placement in Cloud

Jaydeep Das, Sourav Kanti Addya, Soumya K. Ghosh, and Rajkumar Buyya

Abstract Computing resources requirements are increasing with the massive generation of geospatial queries. These queries extract information from a large volume of spatial data. Placement of geospatial queries in virtual machines with minimum resource and energy wastage is a big challenge. Getting query results from mobile locations within a specific time duration is also a major concern. In this work, a bi-objective optimization problem has been formulated to minimize the energy consumption of cloud servers and service processing time. To solve the problem, a crow search based bio-inspired heuristic has been proposed. The proposed algorithm has been compared with traditional First Fit and Best Fit algorithms through simulation, and the obtained results are significantly better than the traditional techniques.

Keywords Geospatial query · Cloud computing · Energy efficiency · Query placement · Optimization

---

J. Das (✉)
Advanced Technology Development Centre, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India
e-mail: jaydeep@iitkgp.ac.in

S. K. Addya · S. K. Ghosh
Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India
e-mail: souravkaddya@cse.iitkgp.ac.in

S. K. Ghosh
e-mail: skg@cse.iitkgp.ac.in

R. Buyya
School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia
e-mail: rbuyya@unimelb.edu.au

D. Mishra et al. (eds.), *Intelligent and Cloud Computing*, Smart Innovation, Systems and Technologies 194,
https://doi.org/10.1007/978-981-15-5971-6_37

1 Introduction

Geospatial Query (GQ) processing is essential in the applications of geographical information systems (GIS), multimedia information systems (MIS), location-based services (LBS), etc. In GQ, the *location* is an essential attribute. To access GIS data, a user can generate GQ from their mobile devices either from a static location or mobile location. In the best case, if the query resolved in a nearby cloud data center (DC), which leads to less communication cost and propagation delay. These metrics will not change for GQs, which are accessed from a static location. Whereas GQs are coming from mobile location users, metrics will change very frequently. GQs extract information from a large volume of spatial data [1]. Therefore, the distribution of GQs in dynamic cloud DCs is challenging [2]. Cloud computing offers a shared pool of huge resources like CPU core, RAM, storage, bandwidth, etc. Virtualization made physical resources available to the user by offering an illusion of a dedicated system. This technique creates multiple virtual machines (VMs) with different configurations of CPU cores, RAM, storage, bandwidth, etc. As per users requirement, the cloud service provider (CSP) offers them required VMs.

GQ load distribution means shifting of GQs to the heavy load VMs to low load VMs. Make a stable cloud environment with equal GQ distribution. While shifting the GQ, keep in mind that the query should be resolved within its timespan and the energy consumption will be minimum. In the literature, very fewer number of works published in GQ placement in the cloud. Lee et al. [3] proposed a spatial indexing technique that works over HBase for big geospatial data. They considered containedIn, intersects, within Distance types of GQ and used GeoLife and SFTaxi Trajectories data points for experiments. In [4], Bai et al. proposed an indexing technique over HBase distributed database using kNN and window query processing algorithms to process huge data objects. A learning technique to resolve GQ in the cloud is discussed in [5]. Akdogan et al. [6] proposed the Voronoi diagram for the efficient processing of a varied range of GQs. The authors have deployed MapReduce-based approach to resolve the reverse nearest neighbor (RNN), maximum reverse nearest neighbor (MaxRNN), and k-nearest neighbor (kNN) queries. GQ resolution on the cloud using geospatial service chains has been discussed in [7].

On the other hand, many algorithms are proposed for load balancing in the cloud. Load balancing is a key factor for GQ placement in cloud servers. Kumar and Sharma [8] proposed an algorithm that is based on a proactive prediction based approach. It predicts future loads based on the past load history and distributed the loads from heavy load VMs to underloaded VMs. Calheiros et al. [9] proposed a workload prediction using the ARIMA model, which helps to get feedback from previous workloads and update the model accordingly. It helps to assign VMs dynamically and maintain the user QoS by reducing the rejection rate and response time. Garg et al. [10] proposed a mechanism to dynamic cloud resource allocation by assigning maximum workload to a DC as per SLA. To maximize the utilization of the cloud DC, they integrated noninteractive and transactional applications. It also reduces the penalty due to less SLA violation as a maximum number of the applications are run in DCs. [11] proposed a framework which schedules tasks decreasing overall energy of cloud data center. Geospatial query resolution is done using cloudlets in [12], and using fog computing in [13].

From the above literature survey, it is observed that most of the works are done separately for GQ processing and load distribution in the cloud environment. These two scenarios have been merged. In this work, GQs are not only resolved in the user-specified time span but also minimize the overall power consumption of the cloud environment due to an efficient GQ load balancing algorithm among the VMs. In this paper, it is assumed that the scheduling of GQs is highly heterogeneous. This work focus on the minimization of GQ resolving time span and energy consumption. The key contributions of this paper are:

- Optimal GQ load distribution with minimal timespan and energy consumption.
- Minimize the service time of the GQ.

The rest of the paper is organized as follows. Categories of the geospatial queries and modeling of processing those queries in the cloud platform are explained in Sect. 2. In Sect. 3, the problem-solution approach using crow search heuristic has been discussed. Performance analysis of the proposed scheme is presented in Sect. 4. Conclusion and future scope of the work are drawn in the last section.

2 System Model

Users generate the GQs through the user interface of web-enabled electronic gadgets, i.e., mobile, laptop, computer, etc. It will be submitted to the Cloud broker. The cloud broker will map the GQ with the existing query types. It is also needed to identify the requirements of geospatial data and geospatial services. There are three types of servers, i.e., Processing Server, Data Server, and Map Server available in the DC. The data server keeps geospatial data. The processing server processes the geospatial data. The map server helps to generate the map on processed geospatial data. After identification, the broker assigns a VM to execute GQ. When the execution is over, it generates the GQ results (GQR), which projected to the user interface. A pictorial view of the system model has shown in Fig. 1.

Assignment of appropriate VM to a GQ is a key feature to distribute the GQ load. Before the assignment of GQ, it is needed to know GQ types, which helps to select VM's specifications during VM to server mapping. The types of GQs [1] are mentioned below.

- **Filter Query**—This type of query[14] filters a particular geometry which presents in another geometry.
- **Within Distance**—It measures whether one geometry or object is present within a particular euclidean distance of another geometry or not.
- **Nearest Neighbor (NN)**—It measures whether geometries is the nearest neighbor of a particular geometry or not.

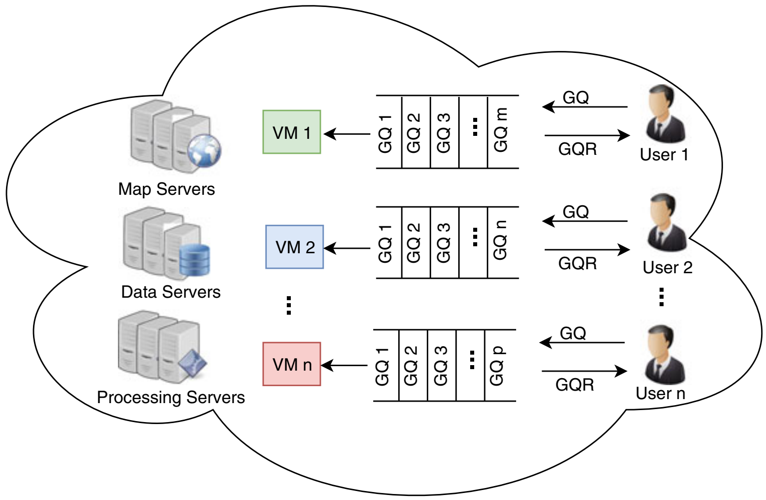


Fig. 1 Geospatial query processing model in cloud

- **Geospatial Join Query**—It compares one layer of geometry with the layers of the other geometries. Geospatial index type (that is, R-tree or Quadtree) must be the same on the geometry column of all the tables involved in the join operation.

2.1 Service Configuration

Cloud DC receives GQs from end-users. To assign GQs in VMs, it has to meet the resource requirements such as CPU, memory, etc. The VM assigns to the GQ if it meets the resource requirements of GQ. The selection of VMs with a number of GQs will be based on its capacity constraints. If requested VM specification is not available, then CSP may assign a higher configured VM. A cloud DC is configured with a large number of servers. VMs are also assigned to the nearby cloud DC of GQ. The nearby cloud DC should fulfill the resource requirement of GQ. Therefore, the optimal mapping of VM to the cloud server is important for maintaining the QoS of GQ services. Scheduling of the GQs is required to place into VMs. After the completion of GQ scheduling, an algorithm is needed for optimal selection of VM among all available VMs.

2.2 VM to Cloud Server Mapping

A suitable mapping of GQs to VMs is required while optimizing overall GQ processing time and power consumption of the system. A geospatial query set

$GQ = \{GQ_1, GQ_2, \ldots, GQ_p\}$ consists of 'p' number of geospatial queries. A VM set $\{vm_1, vm_2, \ldots, vm_q\}$ consists of 'q' VMs. Each GQ has two dimensional (CPU and memory) resource requests. A crow matrix is generated for the GQ requests set. The main focus of the GQ placement (GQP) algorithm is to the continuous optimization of processing time and power consumption. Next, two parameters are represented mathematically.

2.2.1 Processing Time Calculation:

The service time of the GQ processing can be defined as follows:

$$T_s = T_w + T_p \tag{1}$$

where, T_s is Service Time, T_w is Waiting Time in Queue, and T_p is Processing Time in VM. The queue discipline, m/m/1 [15], is considered . All GQ requests will be in a FIFO manner. The arrival rate of GQs is λ and μ will be the service rate. The steady-state probability of 'p' numbers of GQs can be calculated. Thus, the expected number of GQ request in the queue can be assumed, by which the T_w in the queue can also be computed. Similarly, the calculation of T_s can be done.

Now, to process large number of GQs, it needs more VMs. Increment of the number of VMs consumes more energy. If the number of VMs increases, then the energy consumed will be more. This is an NP-Hard problem. Trade-off between the number of VMs generation and overall energy consumption has been done. The relation between processing time in VM (T_p) and energy consumption

$$T_p \propto E_c \tag{2}$$

2.2.2 Power Consumption Calculation:

The total power consumption can be defined as below [15]

$$pwr_i = (pwr_i^{\max} - pwr_i^{\min}) * utz_i + pwr_i^{\min} \tag{3}$$

$pwr_i^{\max}$ and $pwr_i^{\min}$ are the maximum and minimum average power consumed while maximum and minimum utilization is occurred, respectively. utz_i is utilization of vm_i.

2.3 Problem Formulation

In this paper, the GQ placement algorithm is modeled as a bi-objective optimization problem. The objective is to minimize the energy consumption of cloud servers and processing time of GQ. The objective function of the GQP algorithm is mentioned below

$$\text{Minimize} \sum_{c=1}^{n} E_c \quad \text{and} \quad \text{Minimize} \sum_{p=1}^{m} T_p \tag{4}$$

Subject to the three constraints are mentioned below

- ***Assignment constraint***: It assures that the GQs are placed in such VMs where each GQ dimensions are matched with the VM dimensions.
- ***Capacity constraint***: It assures that the total VM requirements of the GQ set should be less than or equal to the total available VMs.
- ***Placement constraint***: It assures to the assignment of a GQ to only one VM which meets the resource requirements along with all dimensions.

3 Crow Search Algorithm for Geospatial Query Placement

To solve the aforementioned bi-objective optimization problem, the crow search algorithm (CSA) [16], has been chosen. CSA is a well-known algorithm that is used to solve many optimization problems. Unlike genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO), chaotic ant swarm (CAS),

Algorithm 1: Crow search algorithm for GQ placement (GQP).

```
Input: GQs from different users
Result: Assign GQs in appropriate VM
Evaluate the position of the GQs.
Initialize the memory of the GQs.
while iter < iter_max do
    Generation of a new assignment of GQs in set GQS''.
    for i = 1 to n do
        Randomly select a VM vm_j that gq_i follows.
        Define the awareness probability AP.
        if r_j ≥ AP_j^iter then
            Check the availability of the resources at VM vm_j.
            Update the position of gq_i in an array GQS'.
        else
            Generate a random VM vm_k with resources.
            Update the position of gq_i in the array GQS'.
        end
    end
    if FL < 1 then
        Set GQS'' = GQS'.
    else
        Perform 'x' random shuffles
        Update GQS''.
    end
    if f(GQS'') is better than f(GQS) then
        Set GQS = GQS''.
    end
end
```

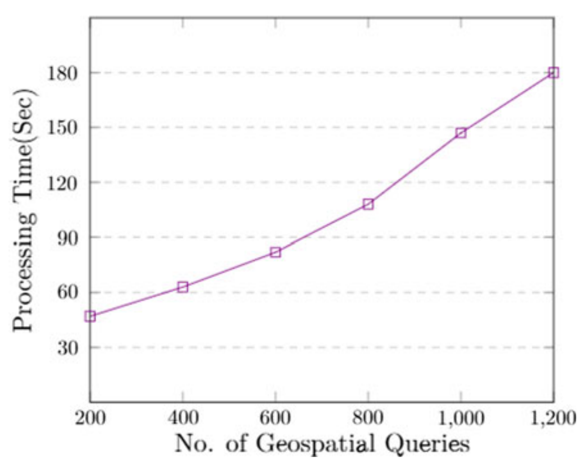
CSA also makes use of a population of seekers to explore the search space. In CSA, the number of adjustable parameters is less (flight length (FL), and awareness probability (AP)) compare to the other optimization algorithms. As adjustable parameters are very difficult to manage. GQP is such an optimization problem where CSA can be used to find its optimal solutions. For GQ placement, GQs are considered as crows, and the best VM for GQ placement is equivalent to an optimal food source. A suitable VM selection is needed as all VMs are uniformly eligible for placing the GQs. As crows search for optimal food sources, similarly, the GQs are searching for appropriate VM for processing. The proposed crow search based algorithm for geospatial query placement into VM is described in Algorithm 1.

4 Performance Evaluation

All experiments are performed in CloudSim 4.0 simulator. The considered number of hosts 100 and each host has 16 processing elements. The number of VMs is varying from 100 to 500, and five types of VMs are considered. These are *Micro VM*(1 Core, 1 GB RAM), *Small VM*(1 Core, 2 GB RAM), *Medium VM*(2 Core, 2 GB RAM), *Large VM*(2 Core, 8 GB RAM), and *xLarge VM*(1 Core, 1 GB RAM). The number of cloudlets (here GQs) is considered within the range of 200−1200. The value of random shuffle is 20, and $iter_{\max}$ is 1000 for keeping the less complexity of the algorithm.

GQs are processed on a first-come, first-serve basis. The number of GQs against processing time graph has been displayed in Fig. 2. Also, the comparison has been done among GQP algorithm with existing Best Fit (BF), First Fit (FF), and Random allocation algorithms in the context of the number of used VM with respect to the number of geospatial queries which are depicted in Fig. 3. It has been observed that

Fig. 2 Processing time versus number of geospatial queries



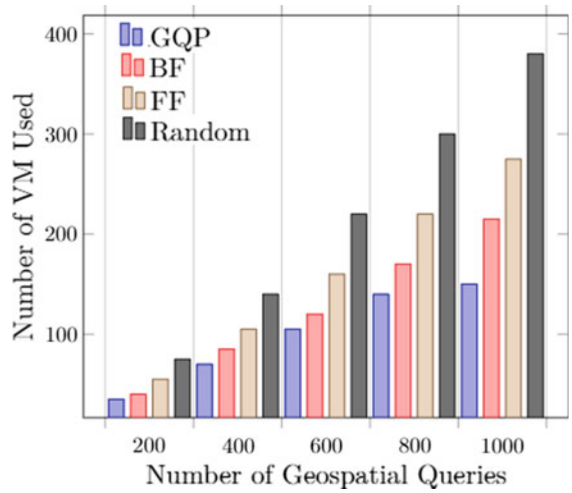


Fig. 3 VMs used by number of GQs for different placement strategies

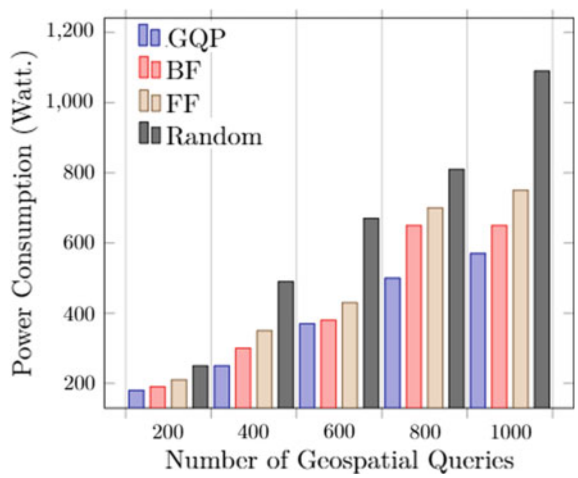


Fig. 4 Variation of power consumption with the number of GQs for different placement strategies

the number of used VMs are increased with the increment of the number of GQs. The number of VMs are used in GQP is lesser than the other three existing algorithms. In FF, first GQ is placed in the first VM. Next GQ will check whether it will fit in the first VM or not. If not, then it moves to the later VM. This moves toward nonoptimal solutions. In the case of FF, the algorithm checks all the VMs capacity and then decides where to move the GQs. This is also moved toward nonoptimal solutions. Figure 4 shows the overall power consumption against the different types of GQs placement strategies. As less number of VMs are used for resolving GQs in the GQP algorithm, this leads to the minimal power consumption in GQP compared to the other three algorithms.

5 Conclusions and Future Work

The processing of GQ from a mobile location in the cloud server is a challenging task. Mainly, GIS data resides in the spatial database in huge volumes. To process a large number of GQ leads to high propagation delay and response time. Also, it causes higher energy consumption in cloud servers. In this work, an optimization problem formulated and solved it using crow search based heuristic. The obtained results are significantly outperforming than traditional FF and BF. An extension of this work, GQ processing in a multi-cloud environment will be done.

References

1. Shekhar, S., Chawla, S.: Spatial Databases: A Tour. Prentice Hall Upper Saddle River, NJ (2003)
2. Yang, C., Huang, Q.: Spatial Cloud Computing: A Practical Approach. CRC Press (2013)
3. Lee, K., Ganti, R.K., Srivatsa, M., Liu, L.: Efficient spatial query processing for big data. In: Proceedings of International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL), pp. 469–472. ACM (2014)
4. Bai, J.W., Wang, J.Z., Huang, J.L.: Spatial query processing on distributed databases. In: Advances in Intelligent Systems and Applications, vol. 1, pp. 251–260. Springer, Berlin (2013)
5. Das, J., Dasgupta, A., Ghosh, S.K., Buyya, R.: A learning technique for vm allocation to resolve geospatial queries. In: Recent Findings in Intelligent Computing Techniques, vol. 1, pp. 577–584. Springer, Berlin (2019)
6. Akdogan, A., Demiryurek, U., Banaei-Kashani, F., Shahabi, C.: Voronoi-based geospatial query processing with mapreduce. In: Proceedings of International Conference on Cloud Computing Technology and Science (CloudCom), pp. 9–16. IEEE (2010)
7. Das, J., Dasgupta, A., Ghosh, S.K., Buyya, R.: A geospatial orchestration framework on cloud for processing user queries. In: Proceedings of International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1–8. IEEE (2016)
8. Kumar, M., Sharma, S.: Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment. Comput. Electr. Eng. **69**, 395–411 (2018)
9. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using arima model and its impact on cloud applications qos. IEEE Trans. Cloud Comput. **3**(4), 449–458 (2015)
10. Garg, S.K., Toosi, A.N., Gopalaiyengar, S.K., Buyya, R.: Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter. J. Netw. Comput. Appl. **45**, 108–120 (2014)
11. Primas, B., Garraghan, P., McKee, D., Summers, J., Xu, J.: A framework and task allocation analysis for infrastructure independent energy-efficient scheduling in cloud data centers. In: Proceedings of International Conference on Cloud Computing Technology and Science (CloudCom), pp. 178–185. IEEE (2017)
12. Das, J., Mukherjee, A., Ghosh, S.K., Buyya, R.: Geo-cloudlet: time and power efficient geospatial query resolution using cloudlet. In: Proceedings of 11th International Conference on Advanced Computing (ICoAC), pp. 180–187. IEEE (2019)
13. Das, J., Mukherjee, A., Ghosh, S.K., Buyya, R.: Spatio-fog: a green and timeliness-oriented fog computing model for geospatial query resolution. Simul. Model. Practice Theory **100**, 102043 (2020)
14. Güting, R.H.: An introduction to spatial database systems. VLDB J. Int. J. Very Large Data Bases **3**(4), 357–399 (1994)

15. Satpathy, A., Addya, S.K., Turuk, A.K., Majhi, B., Sahoo, G.: Crow search based virtual machine placement strategy in cloud data centers with live migration. Comput. Electr. Eng. **69**, 334–350 (2018)
16. Askarzadeh, A.: A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput. Struct. **169**, 1–12 (2016)