International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 11 (2015) pp. 29155-29173

http://www.ripublication.com

Improvement In Load Balancing Technique For Mongodb Clusters Using Data Center Awareness and Ant Colony Optimization

Harpinder Kaur[1], Janpreet Singh[2]
[1] *Student, Department of CSE, Lovely Professional University, Punjab, India*
[2] *Assistant Professor and COD-S, Department of CSE, Lovely Professional University, Punjab, India*

Abstract

Now is the era of cloud computing and related buzzwords are the virtualization, resource sharing, Big Data. With the advent of new technologies, gadgets or simply IOT has enabled the advanced connectivity of devices, systems, and services and with this the data is being produced at an enormous rates from these devices be it form sensors, GPS data, log files from different sources etc. which is mostly unstructured data. With the acquaintance with NoSQL technology MongoDB is extensively used to handle all types of data because of its various advantages as its auto-load balancing technique in which the primary node's read load is decreased by distributing load to the secondary nodes, other feature of MongoDB is its auto sharding technique which works by reducing the load over a node by splitting up data in chunks and migrating it over to other nodes. The present work endeavors to study the role of MongoDB's auto balancing technique. In present work we are going to introduce and examine the performance of balancer of MongoDB for MongoDB clusters in distributed environment and then develop a new load balancing method for better performance in terms of less response time and improved results.

Keywords: Cloud Computing, No SQL, Mongo DB, load balancing, auto-sharding, Ant Colony Optimization, Tag Aware Sharding.

Introduction

Recent years have witnessed a swift burst in web applications that are mushrooming at an astonishing rate per diem. As intricacy of web applications proliferates, the need of data storage is subjected to grow exponentially. Relational DBs are successful in markets for years [1] [10] [11]. These let records to be preserved and retrieved in tabular representation comprising of tuples and columns. However, the tabular

organization has limitations as how to grow horizontally or vertically for the reason that of the requirement of huge magnitude of space requirement for storing all rows. Solution to this question is elucidated by getting the data and separating it into several relations or normalizing data that helps us solving the problem, although, this means data is distributed over the disk and requires several read operations at diverse sectors of the drive to regain the data [10].

SMAC [20], abbreviation for Social, Mobility, Analytics and Cloud is becoming a new savor for business realism. By 2020, as many as hundred billion computing devices will be connected to the World Wide Web and companies will then have to handle fifty times the information than they do at the moment [20]. Because of digitization of business models and processes data is produced at high scale at faster pace over the years. Ginormous amount of users have a habit of requesting same data at a case in point and perhaps need to write particular data at that very same time. Managing of this quantity of user requests is the key concern for firms those are having large scale distributed systems and they want to provide their clients with seamless and highly available services with least response time [29].

In Web2.0 applications, the performance and real time access of database is more vital than ACID properties. The resolution for treating such problems is to use NoSQL databases [4]. The CAP theorem, also referred to as Brewer's theorem, tells that it is not possible for a distributed computer system to concurrently offer all three of the subsequent assurances [19]:

- Consistency: Every node has the exact identical data at the particular instance of interval.
- Availability: The nodes will be able to give response to requests at all intervals if possible.
- Partition tolerance: Works irrespective of a network failure enabling nodes to stay up and running.

There are plentiful advantages and disadvantages for both SQL and NoSQL databases. Developers constantly ponder about as what database will go with their requirements and which amongst them is best so that they may use for their application development. There is no best or direct solution for this query and there is not a single database that would work well for each project. MongoDB has both strengths [20] and weaknesses [21], but on a broad spectrum, it does fairly good and it isn’t having numerous limits and constraints as their other NoSQL counterparts.

The database’s performance is primarily determined by the policies for distributing data and the state of load over cluster for the aim that homogeneously spreads load that can basically help to improve resource usage, maximize throughput and decrease excess load over system. On the other hand, seeing the another characteristic, most LBA (load balancing algorithm) methods for NoSQL are not that proven or are having a smaller amount of adaptability meanwhile it is yet new to marketplace as contrasted to traditional solutions for LB (load balancing) in traditional systems and also the prevalent LB procedures are not appropriate for NoSQL databases [12] [14]. Therefore need of the hour is to work for new strategies to balance cluster load which is more operative, amenable with NoSQL solutions and is valid.

This report proposes an effective load balancing procedure for MongoDB shards using data center awareness and ACO, called MonACO which has led to balancing of load over clusters in a very productive manner. In order to explore the association between prevailing LBA and MongoDB cluster's LB, it is supposed primarily both fields have to be studied individually. So the second and the third sections of this report correspond to our examination of LBA and MongoDB individually and finally improvement in MongoDB Cluster's balancer method using MonACO algorithm. Even though both fields have numerous diverse uses in themselves, emphasize has been given on the central ideas and the readings that were supposed would be valuable for our task.

Mongo DB - A No SQL Database

MongoDB is a potent, flexible, and scalable all-purpose database [17]. This is an open source NoSQL document database, developed by 10gen Company. It was designed to deal with ever growing data storage needs. It is written in C++. It conglomerates the capability to scale out with attributes such as range queries, sorting, secondary indexes, geospatial indexes, and aggregations. This section deals with the major design decisions that made MongoDB what it now is. MongoDB stores data in form of collections. Each collection consists of documents. MongoDB documents are stored in binary format of JSON called BSON. BSON supports binary, Boolean, integer, string, float, date. MongoDB is schemaless [17]. It is easy to store and add new fields to a document or to change the existing structure of the model. MongoDB introduces a new way to distribute collections over multiple nodes this is called Sharding. When nodes comprises of uneven amount of data over clusters, MongoDB automatically reallocates, redistributes the data so that load is equally dispersed across the nodes. It also supports Master-Slave replication. The slave nodes are copies of master node and are used for reads or backups.

Key Mongo DB Features

Mongo DB emphases on speed, power, ease of use and flexibility [22]

- **Speed:** By placing similar data organized in similar related documents, queries can be much quicker than in a traditional database solutions where related data is dispersed to many relations or tables and then requires to be joint later. MongoDB also makes scaling out our database much easier. Autosharding technique allows us to scale our cluster horizontally by adding more machines. It is probable to proliferate capacity without any interruption or downtime, which is very important on the web when load reaches to higher level suddenly and bringing down the website for extended maintenance can cost our business large amounts of revenue [17].
- **Power:** MongoDB offers a many features of traditional RDBMS such as dynamic queries, sorting, secondary indexes, upserts (update if document exists, insert if it doesn't), rich updates, and easy aggregation [7] [8]. This gives us the range of functionality that we used from an RDBMS, with the flexibility and scaling capability that the non-relational model allows [17] [22].

- **Ease of use:** MongoDB works hard to be very stress-free to install, organize, preserve, and use. To this end, MongoDB provides few configuration options, and instead attempts to spontaneously do the "right thing" each and every time possible [9]. This means that MongoDB works right out of the box, and we can dive right into creating our application, in place of wasting a lot of time in fine-tuning vague database conformations.
- F**lexibility:** MongoDB stores data in JSON documents. JSON provides an amusing data model that seamlessly draws to native programming language types, and with the dynamic schema, it easier is to develop our data model than with a structure with imposed schemas such as a RDBMS. This is a typical data insertion command syntax [5][6].

```
/*persons is DB name to which following values are inserted.*/
db.persons.insert (
{
 _id: 1001,
 name: { first: 'Joe', last: 'Florence'},
 birth: new Date ('Sept 05, 1994'),
 langs: [ 'JAVA', 'POLYMER','PHP'],
 awards: [ /*there is no specific schema*/
 { /*this is schemaless structure*/
award: 'McMurphy Award',
by: 'Jane Doe Society'
 },
 {
award: 'National Award of Physics',
by: 'International Physics Organization'
 }
 ]
 }
) //end of insert command.
```

Sharding

Sharding is the method of splitting data up across different machines; the term partitioning can also be sometimes used for describing this concept. When data subset is distributed on each machine, it turns out to likely stock additional data and manage additional load without requiring superior or extra powerful machines, but then again a higher number of less powerful machines are required[23] [24]. Manual sharding can be done with almost any database solutions available. Manual sharding is applicable when an application retains connection to several database servers, each of which are totally autonomous. The application achieves storage of different data on different servers and queries it beside the suitable server to fetch data. This methodology can work efficiently but turns out to be difficult to preserve when increasing or decreasing nodes from the cluster or in the time of need for varying data scatterings or load configurations. MongoDB supports great mechanism called autosharding, which attempts to both abstract away the architecture from the

application and make management of such a system simpler. MongoDB allows our application to overlook the detail that it isn't talking to an individual MongoDB server. On the operational side, MongoDB automates the process of balancing data across shards making it easier to add and remove volume of shards [11] [13] [24].

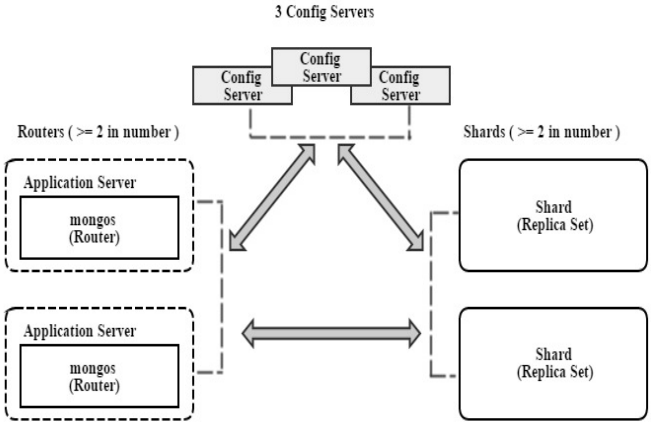

Figure 1: Mongo DB Sharding Architecture

Sharded cluster consists of the following components: shards, query routers and config servers [8] [11] [13].

Shards store the data. Each shard acts as a replica set to provide high availability and data consistency in a production sharded cluster.

Query Routers or mongos instances, provide an interface for client applications and direct operations to the appropriate shard or shards. The query router routes and targets tasks to shards and then return back the desired results to the clienteles. More than one query router can be contained in a sharded cluster so as to share the client request load. Any one query router receives a client requests. Most sharded clusters have many query routers.

Config servers store the cluster's metadata. This data comprises a mapping of the shards to cluster's data set or vice versa. The query router uses this metadata to aim tasks to particular shards. Production sharded clusters consist of exactly 3 config servers. Generally sharding is used to achieve various advantages as it will expand available RAM, maximize present disk space, decrease load on a server, read or write data with better throughput than a single mongod can handle [10].

Tag Aware Sharding

MongoDB provisions a feature in which a range of keys of shards can be tagged to enable to associate that series of range with a shard or cluster of shards. Then these shards will accept all insert operations within that range of tags. The balancer follows tagged range relations, that allows the subsequent organization configurations: [30]

- insulate a particular subgroup of documents on a definite set of shards.
- confirm that the best significant documents exist in shards which are geologically nearby to the application servers.

The balancer transfers chunks of documents in a sharded collection to the shards associated by a tag which consists of a shard key range with a higher value larger than the lower value. Throughout the series of balancing process, if the balancer identifies that whichever chunks do not follow constructed tags, the balancer transfers that chunks to shards related to those specific tags. The cluster takes certain interval to balance the documents amongst the shards after the configuration of tag with range of shard key. This relies upon the partition of chunks and the present distribution of documents in the cluster [30].

Tagging a shard

When we are connected to mongos instance, shard can be tagged using sh.addShardTag() method. A particular shard can consist of several tags, and many shards can also have the identical tag.

sh.addShardTag("s0", "CA");
sh.addShardTag("s1", "CA");
sh.addShardTag("s2", "UK");
sh.addShardTag("s2", "Italy");

Tagging a Range of Shard Key

A range of shard keys can be tagged using *sh.addTagRange()* method once mongos in connected. Several known shard key ranges can have only single allocated tag. Overlap of definite ranges is not allowed, also tagging the similar range greater than one time is also prohibited.

Example

Let's have a collection "empl" in the "e_rec" database, sharded on the field area_zip. The

following steps illustrates how to proceed:

- two ranges of area_zip in Fresno and San Jose have the CA tag
- one range of zip codes in Venice the Italy tag

sh.addTagRange("e_rec.empl", { area_zip: "20001" }, { area_zip: "20291" }, "CA");

sh.addTagRange("e_rec.empl ", { area_zip: "21201" }, { area_zip: "21240" }, "CA");

sh.addTagRange("e_rec.empl ", { area_zip: "87101" }, { area_zip: "84195" }, "Italy");

Removing The Tag

Tag can be deleted from a shard range through removal of the matching document from the tags collection of the config DB. All documents in the tags contains the namespace for the collection which is sharded along with smallest shard key value. The following example removes the CA tag for the range of area_zip within Fresno:

use config

db.tags.remove({ _id: { ns: " e_rec.empl ", min: { area_zip: "20001" }}, tag: "CA" });

Load Balancing

Today, with development of cloud computing, balancing of load over a cluster or server is primary concern for the organizations as it is required to distribute the active native workload equally across all the nodes. It helps to attain a great user contentment and resource consumption ratio by making certain an effective and reasonable provision of every single computing resource.

LB [15][25] is the procedure of scattering of the load amongst several nodes of distributed system so that the improved consumption of resources is achieved in addition to avoidance of a condition where more or less nodes are fully loaded and some nodes are idle or under loaded. Balancing load over cloud delivers a proficient result to numerous concerns residing in cloud computing environment set-up and usage [11]. Balancing is exciting topics in cloud environment. Various methods are to be used to make a system work in a better manner primarily by distributing the loads to the nodes in a balanced manner but because of network crowding, bandwidth consumption etc., but it was not that easy as there were some glitches that happened. These glitches were resolved by more or less of the prevailing methods.

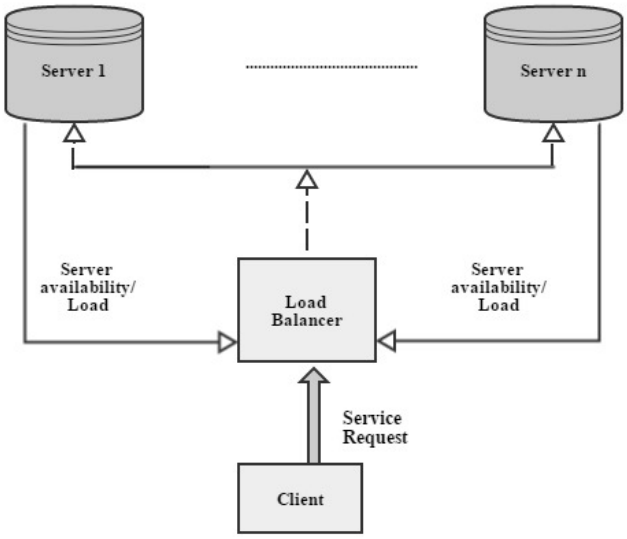

Figure 2: Load Balancing In Distributed Environment

Objectives of Load Balancing

LB is beneficial in various ways for all as for the organization, client and server health. The main objectives of balancing the load are to equally distribute the load on respective node, thus resulting in maximized consumption and minimized total job finishing time [27]. It reduces the reaction time and enhances the resource consumption rate.

Necessity of Load Balancing

LB is considered as a key characteristic of any computing environs which primarily focuses to attain proficient resource planning, supreme consumption of resources and

greater customer contentment, just to assure that none of the particular node is overloaded, therefore we can establish that for refining the overall performance of the system, appropriate balancing of load is required [15].

Design Issues in LBA

Various design concerns for LBA. Numerous problems are considered while the development of a good LBA, such as deciding rules for load approximation, process transmission, static info interchange, place, precedence assignment, and relocation limitation [17] [15].

The load approximation policy governs how to approximate the job-load of a node in a distributed network [27]. The taxonomy of load approximation policies are determining the number of procedures, processes running on a machine, taking CPU demanding time, the *process transmission policy* chooses whether the process can be completed locally or there is a need for remote implementation. The need of the hour is to choose a policy that specifies if a node is greatly or lightly loaded as compared to predefined value called the threshold policy. The kinds of threshold policy are single-level threshold, and two-level threshold. *Static info interchange* policy defines how the system-wide load statistics can be swapped among the nodes. The LBA uses the state info interchange strategies as periodic broadcast, broadcast upon state change, interchange of state info when demanded, and interchange by polling [29]. *Location policy* decides the election of a target node during process transmission. The *precedence assignment* strategy determines the precedence of implementation of a set of native and distant processes on a specific node. The *relocation limiting* policy regulates the number of times a process is allowed to be transferred [29] [2].

Swarm Intelligence – Ant Colony Optimisation

Swarm Intelligence can be defined as the property of a system in which the co-operative actions of (ingenuous) agents working together nearby their surroundings tend to result in emergence of coherent well-designed comprehensive patterns. Swarm Intelligence offers a foundation by which the exploration of shared or co-operative (or distributed) problem solving becomes possible devoid of central control or the establishment of a global model. This notion is in deployment in field of AI [32]. This is a moderately new methodology to problem solving which earns motivation from social behaviors of creatures of insect family besides other faunae. Precisely, ants have motivated many approaches and techniques amongst which the maximum considered and the greatest effective is the metaheuristic optimization method popularly called Ant colony Optimization (ACO). ACO is inspired from the foraging activities of certain ant class or species. These ants tend to deposit a chemical called pheromone during the journey in order to mark certain productive trail that must be followed or travelled by other fellows of the colony. ACO exploits a related technique for resolving optimization problems [33]. It is a model for the design of metaheuristic set of rules for combinatorial optimization problems. In the early 1990s, ant colony optimization (ACO) [34] [35] was presented by M. Dorigo and his co-mates as a unique nature-motivated metaheuristic for the resolution of hard Combinatorial

Optimization Problems (COP) and then, several varied variations of the base norm have been developed for various fields of study. The vital feature of ACO processes is the grouping of a-priori statistics of the construction of a favorable resolution with a posteriori statistics of the system of formerly gathered worthy resolutions and solutions. This has till now used for travelling salesman problem [36], scheduling, routing as for vehicle routing problems [37] etc.

Behavior of Ants

Distinct ants are behaviorally much naïve insects. They possess a very restricted remembrance and display specifically distinct behavior which seems to have a large arbitrary factor. Working in a group however, these tiny ants are able to accomplish diversity of complex tasks with great consistency and dependability [33].

How Ants Work

ACO's inspirational foundation is foraging manners of real ants. Ants acting as agents primarily discover and roam about the surrounding area of their nest in an arbitrary fashion just to search for food. Once an ant discovers a source for food, it assesses the amount and the superiority of the food and brings certain amount of it back to its nest [33] [36]. For the period of the round trip, the ant leaves behind a chemical pheromone track onto the path travelled by it. The amount of pheromone dropped or deposited, that might be dependent onto the amount and excellence or validity of the food, will direct other ants from nest to the source of food.

More pheromone, more qualitative food. Otherwise due to less travelled path, pheromone evaporates too. Reliant on the genre of species, the ants might tend to deposit pheromone trails while on their either journey i.e. either from nest to food, or from food to nest, or also while traversing any of the both direction. Moreover these observe and follow these traces of pheromone with a trustworthiness that is a function of the power of trail, along with other factors. Ants leave behind pheromones when they walk after halting for a moment and slightly pressing their gaster that contains the pheromone discharging gland, onto ground. The power of the trail that is laid is dependent at the rate with which pheromone is deposited, also the aggregate per deposit. [39] In the meantime when pheromones get evaporated or got dispersed away, the power of the trail gets dependent of the original power and 5the timestamp since that trail was deposited by ant.

Let us take into consideration a network in which ants can traverse among diverse nodes. Using deposition of pheromone, the ant 'k' currently at node 'i' will decide to go to another node is given by the probability as shown [34]:

$$P_{ij}^{k} = \frac{(\tau_{ij}^{k})^{\alpha} \times (\eta_{ij}^{k})^{\beta}}{\sum_{l \in N_i^k} (\tau_{il}^{k})^{\alpha} \times (\eta_{il}^{k})^{\beta}}$$

This is possible only $j \in N_i^k$ if. Otherwise P_{ij}^{k} will be equal to 0. In the equation, level of pheromone is symbolized by τ_{ij}^{k} as is the case with real ants, possibility of an ant taking up a particular path increases with increased deposition of pheromone at that path. The sum in the denominator is done to take into account all

the possible selections from neighboring nodes in the set N_i^k where the ant currently is at i[th] node. The factors $\alpha, \beta, \eta_{ij}^k$ are usually application dependent; η_{ij}^k is the heuristic info, and the value of α, β assess the significance of heuristic and pheromone values. When

$\beta = 0, (\eta_{ij}^k)^\beta = 1$

then the probability is dependent only upon levels of pheromone in contrast, when α=0, the probability relies only upon heuristic values i.e., closest node from the current node which will have the maximum probability of it getting chosen [33] [36].

The pheromone intensities on path from node i to j, can get evaporated with a percent ρ (also called rate of evaporation) [33]:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}$$

This equation is valid for $0 \le \rho < 1$. Consequently the fresh pheromone intensities are updated with some other pheromone left by the ants who just traversed that path [33]:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k$$

$$\Delta\tau_{ij}^k = \frac{1}{C^k}$$

Here C^k is the related to cost for ant ‘k’ when selecting this path to traverse.

Literature Review

Nyati S.S., Pawar S, Ingle R. [3] have very cautiously explained few NoSQL unstructured databases and additional performance inspection of these considered databases was offered which was completed on basis of several benchmarks The contrast is shown between MongoDB and MySQL over various benchmark and the results showed MongoDB is very quicker in inserting the data as well as quicker in searching.

BANKER K [10]: Chapters 1-6 gave out MongoDB's background, strategies and situations in which MongoDB is used, due to what reasons, MongoDB is so unique, than its other NoSQL counterparts. Predominantly established for scalability necessities of applications, MongoDB provisions for dynamic queries and secondary indexes; faster atomic updates and intricate aggregations; and provision for sharding for horizontal scale along with replication along with automatic failover.

Liu Y, Wang Y, Jin Y [12]: Authors offered some information about rules and deployment approaches of auto sharding in MongoDB, and then set forth an improved form of their procedure to resolute the trouble of unbalanced scattering of data in auto-sharding based on occurrence of data operations. The advanced balancing outline

efficaciously balances the data among shards, and increases the cluster's simultaneous writing and reading performance.

Huang, Chao-Wen; Hu, Wan-Hsun; Shih CC;Lin BT;Cheng CW [14]: In this work, sharding characteristics of MongoDB and the use of on-request characteristics of cloud computing were enlightened to offer a key to virtualized auto-scale the database that will meet the SLA necessities. In the beginning, the author used auto-scaling mechanism of route server in MongoDB system. The investigational outcomes shown that the average reaction time of auto-scaling DB was 4.3 sec and non-scaling DB was 7.1 seconds. Furthermore, they also prototyped a shard data migration algorithm that lead to decreased impact while migrating data to new virtual machine. The auto-scaling DB solution used the algorithm to determine how many additional VMs to be added and which data has to be transferred to those afresh added VM.

Wang XL, Chen H, Wang Z [15]: This paper emphasized the preference towards MongoDB for its expertise in automatic LB system and the auto sharding technique. The authors have focused on the working of automatic LB of MongoDB and recommended a dynamic LB method based on heat diffusion from cluster with much less cost. The suggested dynamic LB algorithm based on heat diffusion is significant for improved results.

Sim K.M. and Sun W.H. [41] have presented a thorough analysis of ACO and traditional routing algorithms. Solution to the problem of stagnation was discussed and measures were suggested to avoid it from occuring.The solution includes evaporation of pheromone, aging of trail of pheromone, pheromone smoothing and limiting, privileged pheromone laying and pheromone-heuristic control. Multiple ant colony optimization in load-balancing was discussed further.

Ali A.D and Belal M.A.[42] proposed an ACO for distributed environment. This includes updating of information dynamically at each movement of ant. Multiple colonies approach was used in order to avoiding the ant from same colony visiting same route at initial stage. This helped in exploration of all possible ways to gather information of nodes that were idle to balance load.

Dorigo M., Stützle T. [35] have extensively provided thorough explanation about the ant system ant their foraging behavior, how they approach to solve the problem. The presented book majorly defines the interpretation of witnessed ant behavior into operational optimization procedures. The ACO metaheuristic at that time is presented and observed in the all-purpose general framework of COP. Subsequently, a comprehensive explanation and guide for entire leading ACO procedures is given and then existing theoretic results are elaborated. The book assesses ACO uses that are currently in usage, comprising routing, task-assignment, arrangement, and bioinformatics difficulties. Description about a network routing known as ANTNET is presented which is based on ACO. The authors accomplish the book by briefing out the advancement in the respective fields and drawing base for upcoming research guidelines.

Present Work

Problem Formulation

While working with NoSQL database MongoDB, amassed amount of user requests are controlled by routing read requests to secondary nodes of replica sets and write requests to the primaries of replica set. Thus if all secondaries are busy serving requests, there may be a chance that the requests are queued for a long time. In order to solve this problem all the user requests are efficiently routed to under loaded shards based on user request parameters and tag based sharding of shards.

Objectives and Methodology

This research work is basically intended to improve the performance of MongoDB sharded clusters in distributed environs that is handling enormous volume of user requirements at a particular instance. The work is done so that the scalability and availability of data is not compromised. In order to meet the requirements of scalability, sharding is done and for availability, replica sets are used inside shards. So here, a custom data center aware algorithm based on ACO and MongoDB's tag-aware sharding is developed for workload balancing over clusters. All the clusters will initiate the algorithm which will monitor the shard status. If a shard is about to reach predefined value defined for each shard as threshold, developed algorithm will start its work and nearest request specific shard will be found and then the user request will be route to that shard using mongos process to balance the load over cluster.

Adaptation of ACO jargon to MongoDB

In this section, adaption of ACO metaphor in MongoDB's metaphor is discussed.

Food Type: The set of key:value paired documents in shards who have values in them that are located in specified range of tags, determine food type.

Food Resource: The set of all key: value pairs stored in each sharded collection in each replica set of shard is matched with food resource. Every food resource contains many kinds of food types. Each node in network is assumed to be food resource.

Pheromone: Path to be traversed is determined with the help of pheromone table updated each time ant traverse.

Pheromone Update: Updates are made to pheromone table when concentration of new node found is larger than previously found node in order to approach for best path selection for query to route using mongos process.

Algorithm- MonACO

STEP I: *initiate all 3 tag-based sharded shards and their respective replica sets at an interval of 5 sec.*

STEP II: *start 3 config servers*.

STEP III: *wait for 60 sec so that all mongod, mongos are up and running*.

STEP IV: *side by side MonACO starts working*.

Initiate pheromone on each node to be 0.

setupMAnts(): ants set to discover all random nodes for food discovery.

moveMAnts(): move to next node and update nodeTable.

updateMAntTrails(): evaporation takes place in this step if node is least visited.
updateMBest(): calculate the path length and update the best found so far.

STEP V: from *updateMBest()* route the user request to the best found shard node for serving request.

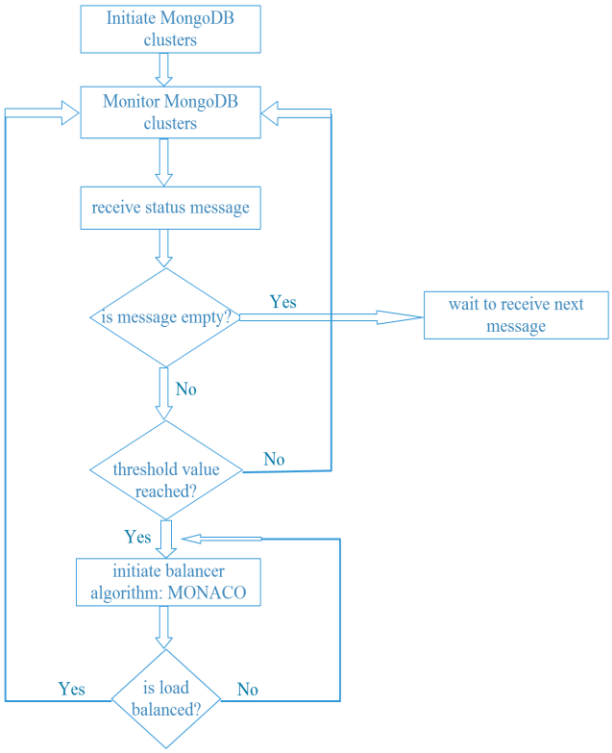

Figure 3: Proposed Methodology For Balancing Load of Mongodb Clusters

Fig-3 shows flow of proposed MonACO algorithm for balancing load across clusters. Load is number of user request coming and overwhelming the system. So to balance and maintain performance of MongoDB clusters, algorithm is made to do its job as depicted in Fig-4. MonACO step initiation step in Fig-3 (marked with ****) is shown in Fig-4 to elaborate its steps.

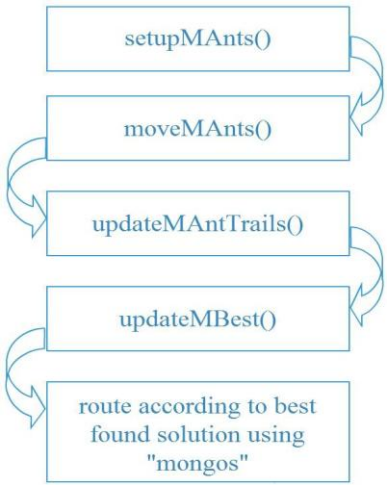

Figure 4: Proposed Mon ACO

Design Methodology

The implementation was done using a platform running on Windows 7 Ultimate OS, having i5-2430M CPU @ 2.4GHz and 4GB RAM. MongoDB 2.6.4 is used. Distributed environment was simulated using different ports for mongod, mongos processes and mongod processes are started that will work as replica sets of shards. Code implementation is done in Java using NetBeans IDE. MongoDB jar file helped in working with MongoDB using JAVA. The process starts by initiation of replica sets of shards. "mongod" processes are started. For this port numbers are entered. Primary node is selected for each replica set. And press Initiate Replica Sets button. Wait for about 15 seconds so that each replica set is up and running. For testing the performance of clusters, first monitoring is done without use of proposed algorithm. Then after that monitoring is done while using the proposed MonACO. MonACO is simulated by passing a file consisting of shard IDs and distances of replica sets from each other.

```
E:\NetBeansProjects\param\src>java MonACO mongoDis.txt
Best shardID for routing request : 79.0
```

Figure 5: Mon ACO Result

The best shard ID is returned. This is further given to mongos process. And the request is routed to returned best solution of shard. This method also reduces amount of metadata that mongos has. Thus improving overall performance of cluster.

Results and Discussions

In present chapter anticipated results are discussed that were expected after the implementation work is complete. According to the present work, when the algorithm has done its part the cluster has become balanced, it is neither be overloaded nor

under loaded instead just is balanced. Consequently, thus response time is reduced and eventually the cluster performance is also improved.

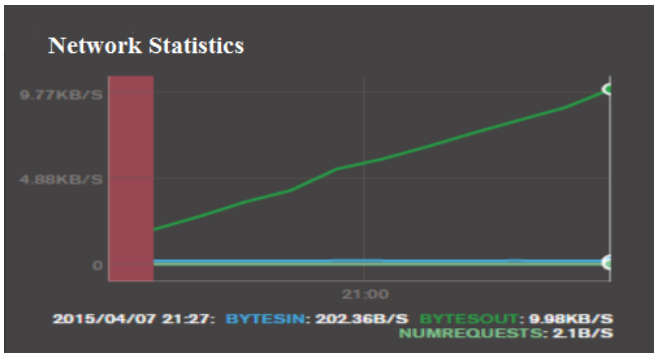

Figure 6: Network Graph Obtained Without Using Monaco

This is network graph, Fig-6 is obtained while working with MongoDB cluster simulation with replica sets. The graph depicts network load for maximum queries coming to MongoDB. The balancer starts balancing load and it becomes quiet balanced. The various parameters observed are: number of requests served, NumRequests at 2.1B/s, BytesIn at 202.36 B/s, BytesOut at 9.98KB/s.

This another network graph, Fig-7 is obtained while working with one module of proposed algorithm with MongoDB clusters. The graph portrays network load for maximum queries coming to MongoDB. This is observed over a time of 12 hours with granularity of 5 minutes. The custom balancer MonACO does balancing of load. Resultant parameters observed as with algorithm's operation elucidates Num Requests: 2.75 B/s, Bytes In: 236.16 B/s, BytesOut : 11.15 KB/s.

This improvement proves really advantageous in distributed environment where user requests are being growing at an enormous rates and to cater their needs balancing proves to be savior.

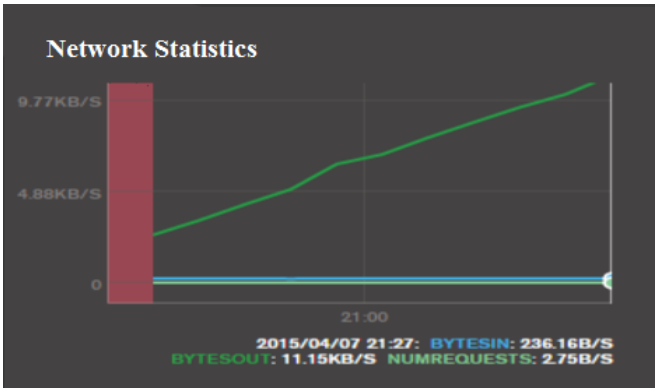

Figure 7: Network Graph Obtained With Using Monaco Algorithm

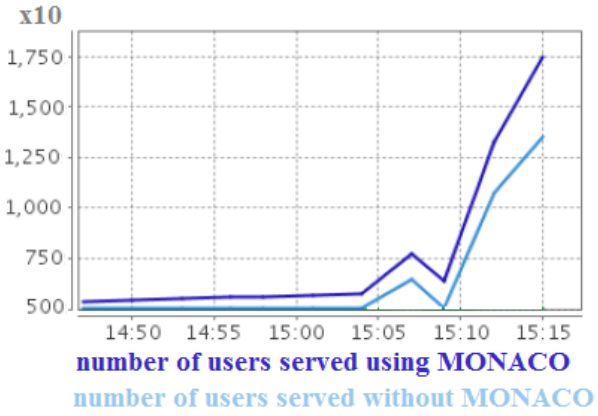

Figure 8: Number of Requests Handled After and Before The Use of Monaco

Conclusion and Future Scope

The load-balancing of servers is vital for storage applications that are mostly read intensive. Traditional balancing methods cannot be relied upon for distributed environment like for MongoDB's case under consideration. So efficient solution for balancing load over distributed MongoDB clusters is presented based on datacenter awareness and ACO which has eventually resulted in improvement of the shard performance when huge amount of load arises. There is a difference of (2.75-2.1) 2.65B/s between both methods with respect to NumRequests. Also the (17500-13801) 3699 more of users being served while working with MonACO as compared to when not working with MonACO The present work will be designed to work in case of cloud where there is huge number of users being handled. MonACO approach will then can serve as policy for DBaaS to effectively manage cloud load over a wide range of geographic area, this will be based on MongoDB's geographic indexing.

Acknowledgement

I owe my gratefulness to all those who have helped in making this dissertation possible so far. First and foremost, I'd offer my sincere thanks to my mentor, Mr. Janpreet Singh, who has assisted me during the course of my study by his perseverance and information while allowing me the area to explore in my own approach.

I am grateful to Mr. Nitin Kumar for his motivation and applied practical advice. I am also thankful to him for understanding my reports, remarking on my interpretations and assisting me to understand and improve my concepts.

I'd also like to acknowledge Mr. Rohit Dhand for frequent negotiations and discussions on related areas that supported me in progress of my work and awareness in the region.

Several friends have aided me stay comprehensive during the hard months. Their livelihood and concern help me out to conquer obstacles and stay focused and

encouraged on the way to my study process. I highly value their connection and I intensely admire their confidence in me. Most markedly, not a jiff of this would have been possible without the love and determination of my kinfolk. My kinfolk has been an infinite source of love, regard, care and positivity all these months. I'd like to express my thanks to my kinfolk.

References

[1] Sosinsky B., Cloud Computing Bible, Indianapolis, Indiana: Wiley Publishing, Inc., 2011, p. 109.

[2] "NOSQL DATABASES EXPLAINED," [Online]. Available: http://www.mongodb.com/nosql-explained.

[3] Nyati S.S., Pawar S, Ingle R., "Performance Evaluation of Unstructured NoSQL data," in 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Mysore, 2013.

[4] Brust A., Dash J., "RDBMS vs. NoSQL: How do you pick?," 18 September 2013.[Online]. Available: http://www.zdnet.com/rdbms-vs-nosql-how-do-you-pick-7000020803/.

[5] Kaur K, Rani R, "Modeling and querying data in NoSQL databases," in IEEE International Conference on Big Data, Silicon Valley, CA, 2013.

[6] Aggarwal R, Arora R, "Modeling and Querying Data in MongoDB," International Journal of Scientific & Engineering Research, vol. Volume 4, no. Issue 7, pp. 141-145, July-2013.

[7] Alexandru B., Florin R., Laura I. A., "MongoDB vs Oracle - database comparison," in Third International Conference on Emerging Intelligent Data and Web Technologies, Bucharest, 2012.

[8] Coe, B., "To MongoDB, or Not to MongoDB," [Online]. Available: http://www.codemag.com/Article/1309051.

[9] A. R. Arora R, "An Algorithm for Transformation of Data from MySQL to NoSQL (MongoDB)," International Journal of Advanced Studies in Computer Science and Engineering (IJASCSE), vol. 2, no. 1, pp. 6-12, 31 July 2013.

[10] BANKER K, Mongo DB in Action, Shelter Island, NY: Manning Publications Co., 2012, pp. 3-126.

[11] Chodorow K, Scaling Mongo DB, L. M, Ed., Sebastopol, CA: O'Reilly Media, Inc., 2011, pp. 1-45.

[12] Liu Y, Wang Y,Jin Y, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," in The 7th International Conference on Computer Science & Education, Melbourne, VIC, 2012.

[13] Zugic G, "Selecting a MongoDB Shard Key," 29 May 2014. [Online]. Available: https://goranzugic.wordpress.com/2014/05/29/selecting-a-mongodb-shard-key/.

[14] Huang, Chao-Wen; Hu, Wan-Hsun; Shih CC;Lin BT;Cheng CW, "The improvement of auto-scaling mechanism for distributed database - A case

study for MongoDB," in Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific, Hiroshima, Japan, 2013.

[15] Wang XL, Chen H, Wang Z, "Research on Improvement of Dynamic Load Balancing in MongoDB," in Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference, Chengdu, 2013.

[16] "SMAC," [Online]. Available: http://www.cognizant.com/smac. Kristina Chodorow, Mongo DB: The Definitive Guide, Sebastopol, CA: O'Reilly Media, 2013, pp. 6-232.

[17] Raj Kumar Buyya, "Cloud Computing: Principles and Paradigms", Indianapolis, Indiana: Wiley Publishing Inc., 2011.

[18] P. Kushwah, "A Survey on Load balancing Techniques Using ACO Algorithm," (IJCSIT) International Journal of Computer Science and Information Technologies, vol. 5, pp. 6310-6314, 2014.

[19] "SMAC," Cognizant Technology Solutions, [Online]. Available: http://www.cognizant.com/smac.

[20] "CAP Theorem," Foundation DB, [Online]. Available: https:// foundationdb. com/ keyvalue-store/white-papers/the-cap-theorem.

[21] H. C. ,. Z. W. XiaoLin Wang, "Research on Improvement of Dynamic Load Balancing in MongoDB," in Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference, 2013.

[22] "MongoDB - Advantages",[Online]. Available: http://www.tutorialspoint. com/mongodb/mongodb_advantages.htm.

[23] "Two Reasons You Shouldn't Use MongoDB",[Online]. Available: http://www.databaseskill.com/3091951/.

[24] "Introduction to MongoDB", [Online]. Available: http://www.mongodb. org/about/introduction/.

[25] "Sharding Introduction", [Online]. Available: http://docs.mongodb. org/manual/core/sharding-introduction/.

[26] "Sharded Collection Balancing", [Online]. Available: http://docs. mongodb.org/manual/core/sharding-balancing/.

[27] [Online]. Available: http://searchnetworking.techtarget.com/definition/ load balancing.

[28] R. Wanker. [Online]. Available: http://dcis.uohyd.ernet.in/~wankarcs/ index_files/pdf/pc-04-02.pdf.

[29] "Tag Aware Sharding," MongoDB Inc., [Online]. Available: http://docs. mongodb.org/manual/core/tag-aware-sharding/.

[30] P. Werstein, "Load Balancing in a Cluster Computer," in Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06, Taipei, 2006.

[31] "Swarm Intelligence", [Online]. Available: https://www.wikiwand. com/en/Swarm_intelligence.

[32] M. B. a. T. S. Marco Dorigo, Ant Colony Optimization-Artificial Ants as a Computational Intelligence Technique, IEEE, 2006.

[33] V. M. A. C. M. Dorigo, “Ant system: optimization by a colony of cooperating agents,” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, pp. 29 - 41, February 1996.

[34] T. S. M. Dorigo, Ant Colony Optimization, Cambridge, MA: MIT Press, 2004.

[35] M. D. L.M. Gambardella, “Solving Symmetric and Asymmetric TSPs by,” Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96, pp. 622-627, 1996.

[36] R. H. C. S. B. Bullnheimer, “Applying the ant system to the vehicle routing problem,” Voss S., Martello S., Osman I.H., Roucairol C. (eds.) MetaHeuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 285-296, 1996.

[37] W. K. Macura, “Ant Colony Algorithm,” [Online]. Available:http://mathworld.wolfram.com/AntColonyAlgorithm.html.

[38] Y.-T. Hsiao “Computer network load-balancing and routing by ant colony optimization,” in 12th IEEE International Conference on Networks, 2004. (ICON2004), 2004.

[39] J. Brownlee, Clever Algorithms: Nature-Inspired Programming Recipes, 2011.

[40] K. M. S. a. W. H. Sun, "Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 33, no. 5, pp. 560 - 572, 2003.

[41] A.-D. A. a. M. A. Belal, "Multiple Ant Colonies Optimization for Load Balancing in Distributed Systems," in *International Conference On Information and Communication Technology and Accessibility*, Hammamet, Tunisia, 2007.