Partitioning of Query Processing in Distributed Database System to Improve Throughput.

Ms. Pratibha B. Patil[1] , Mr. Rahul P. Mirajkar[2], Ms.Neeta B. Patil[3]

[1]Student, Department of Computer Science and Engineering, Bharati Vidyapeeth's College of Engineering, Kolhapur, Maharashtra, India

[2]Assistant Professor, Department of Computer Science and Engineering, Bharati Vidyapeeth's College of Engineering, Kolhapur, Maharashtra, India

[3]Student, Department of Computer Science and Engineering, Bharati Vidyapeeth's College of Engineering, Kolhapur, Maharashtra, India

---***---

Abstract - *Query processing in distributed system calls for the transmission of records among computers in community. The arrangement of statistics transmission and local information processing is known as distribution strategy for a query. Two cost measures reaction time and total time are used to judge the great of distribution method. Numerous algorithms are used that derive distribution strategies that have minimal reaction time and minimum overall time. The optimal algorithms are used as a foundation to increase a general query processing algorithms. In this article, we introduce using graph partitioning to partition the task of query to optimize the throughput on distributed structures. New heuristic set of rules we've advanced, the congestion avoidance partitioning (CA) algorithm to optimize the throughput and query execution (parallel) algorithm developed for concurrent execution of query in distributed database system. We evaluate the query processing device with and without partitioning algorithm to analyze throughput end result.*

Key Words: **Computer Network, Distributed database, query processing, graph partitioning, concurrent execution.**

1. INTRODUCTION

In recent years, with the development of computer network and database technology, dispensed database is more and more extensively used; with the increasing application, facts queries are increasingly more complicated, the performance requests are an increasing number of excessive, so query processing is a key trouble of the dispensed database device.

In a distributed database surroundings, data stored at exclusive sites linked through community. A distributed database management system (DDBMS) aid advent and maintenance of disbursed database. The studies literature proposes a huge form of query optimization algorithms and overviews on diverse query optimization techniques for distributed database management system[5,6,7]. However, these overviews do no longer try and increase a model of query optimization that explains and gives the algorithms in a uniform manner. This knowledge in case we want to exchange or enlarge current algorithms to conform them to new necessities. In this research we take into account query processing algorithms for an allotted Database machine.

To gain good performance for query processing on parallel structures with shared reminiscence, above mentioned initiatives have deployed sever scheduling strategies. The primary trend is to maximize the parallelism among all cores by way of mapping computational nodes of the queries into individual cores. Special graph partitioning strategies have been taken on to provide load stability so far both regardless the communication value or bear in mind it simplest with a low priority[1,2,3].

On distributed structures in which communication cost has an large effect on performance, those strategies are now not appropriate. In this paper, we advocate a graph partitioning method especially suitable to optimize the throughput of preferred query processing on dispensed database platforms. A heuristic and approximation algorithms seeking to divide a graph into separated partitions to optimize a goal. The most common and well investigated objective is to equalize the scale of partitioning while minimizing the entire cuts between them [1,4,5].

For query processing, the throughput is decided now not simply via the workload on each partition, however additionally by the communication fee among each pair of partitions. it's far even extra complicated when the dispensed platform and the verbal exchange bandwidth isn't uniform among them. We have got developed a set of rules, called Congestion Avoidance (CA) that narrows the hunt area. Rather than examining all opportunities, this method concentrates on moves around congestion points in which the throughput is dimmed[1,8,9,10,11].

2. LITERATURE REVIEW

Many non-standard graph partitioning and query processing techniques have been proposed in the literature.

Raimund Kirner(2015)[1] proposed two novel heuristic graph partitioning techniques to partition the workload of circulate packages to optimise throughput on heterogeneous dispensed systems and comparison of both KLA and CA with the prevalent meta-heuristic Simulated Annealing (SA). All three techniques attain comparable throughput effects for maximum instances, however with substantial differences in calculation time. For small graphs KLA is faster than SA, however KLA is slower for large graphs. CA however is usually orders of magnitudes quicker than both KLA and SA, even for big graphs. This makes CA probably useful for re-partitioning of systems at some point of runtime.

Khandekar, S. Rao, and U. Vazirani(2006)[2] proposed an algorithms for graph partitioning problems that use single commodity flows. Algorithms iteratively route flows throughout cuts within the graph and embed an expander with congestion within times the most appropriate. In case of the sparsest reduce trouble, the union of the flows, in truth, corresponds to an embedding of a whole graph.

B. L. Chamberlain(1998)[3] proposed graph partitioning algorithms used for parallel computing, with an emphasis at the problem of distributing workloads for parallel computations. Geometric, structural, and refinement based algorithms are defined and contrasted. similarly, multilevel partitioning strategies and issues associated with parallel partitioning are addressed. All algorithms are evaluated qualitatively in phrases of their execution speed and capability to generate partitions with small separators.

Fan Yuanyuan, Mi Xifeng(2010)[4] researched on query optimization technology, based on some of optimization algorithms commonly utilized in dispensed query, a brand new set of rules is designed, and experiments show that this set of rules can extensively reduce the amount of intermediate end result records, efficiently reduce the network communication value, to enhance the optimization performance.

Kunal Jamsutkar (2013)[5] proposed traditional technique of processing a query in a relational DBMS is to parse the SQL. Announcement and convey a relative calculus like logical illustration of the question, and so to invoke question optimizer, which generates query plan. The query plan is fed into an execution engine that at once executes it, typically with little or no runtime choice making.

B.M. Monjurul Alom, Frans Henskens and Michael Hannaford (2009)[6] researched The (FRS) fragment and replicate method allows parallel processing of a query. Fragments and replicate(FRS) techniques aren't applicable for processing disbursed queries; in which all of the non fragmented relations are referenced by a query. A new placement dependency algorithm is offered to improve FRS strategy, but the trouble of this algorithm, when the quantity of web sites increases is that, the statistics distribution becomes sparse and the probabilities that the corresponding fragments of the referenced members of the family are located at the equal website online turn out to be smaller. for this reason the algorithm will become much less beneficial. any other drawback of this algorithm is that the average improvement decreases as the number of referenced relations will increase although the lower is as a substitute gradual.

Peter M. G. Aperes, Alan R. Hevner, and S. Bing Yao(1982)[7] proposed two versions of set of rules GENERAL to reduce response time of a processing approach, parallel statistics transmissions are emphasized by the use of set of rules PARALLEL and technique response. algorithm general(reaction time) may be proved to derive minimal response time techniques beneath the assumption of characteristic independence within question relations. To reduce the whole time of a processing strategy, serial time transmissions are emphasized through using algorithm SERIAL and system general in set of rules standard (overall time).

Donnald Kossman(2000)[8] proposed the state of the art of query process for distributed information and data system. He also presented the "textbook" design for distributed query process and a series of techniques that square measure significantly helpful for distributed information systems. These techniques embrace special be a part of techniques to take advantage of intra query similarity, techniques to cut back communication prices, and techniques to take advantage of caching and replication of information.

4. PROPOSED SYSTEM

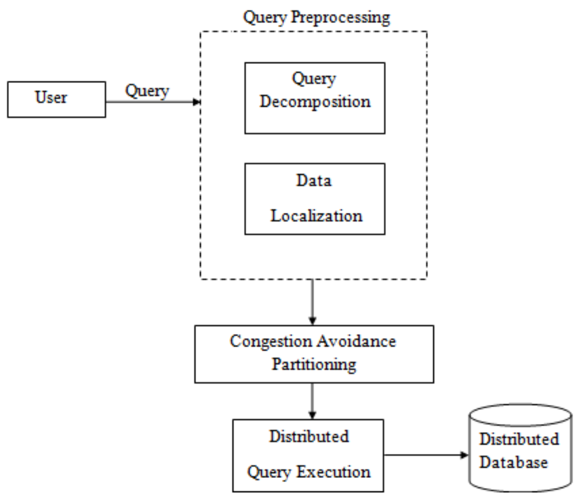


Figure 1. Schematic of distributed query execution using partitioning algorithm.

1. **Query:**

In figure 1 query is an input given from user for preprocessing, A database query is the instructing a DBMS to replace or retrieve unique records is executed via diverse low degree operations.

2. **Query Preprocessing:**

There are two levels that query passes through at some stage in preprocessing of that query.

A. Query Decomposition
B. Localization

Query Decomposition: At this stage user given query as an input and got output as fragmentation of those query, reveals the attribute included within the query and fragments it for processing or localization.

Localization: At this stage fragmented query or attributes are input and its related hosts and mapped tables are as output. It localizes decomposed attributes to its associated servers or host and its mapped tables wherein the ones are available.

3. **Partitioning:**

At this stage when queries are divided in sub queries then apply the graph partitioning algorithm (Congestion Avoidance (CA)) on the task of query which incorporates executable query to maintain a strategic distance from the system activity for expanding the throughput of query processing.

Congestion Avoidance Partitioning Algorithm: The Congestion Avoidance (CA) that limits the inquiry space. Rather than analyzing all potential outcomes, this strategy focuses on moves around blockage focuses where the throughput is decreased.

The Congestion Avoidance (CA) partitioning algorithm starts with an underlying mapping setup furthermore, rehashes a heuristic go until the throughput achieves a locally ideal esteem. CA pass does not inspect all conceivable move operations, however concentrates on as it were those around the blockage point distinguished by assessing the throughput formula of Equation 5. Inside each pass, the CA recognizes the blockage purpose of the present mapping arrangement and tries move operations that conceivably enhance the throughput.

From Equation 5 which is described in [1] for stream programs, in this article we have replaced stream programs in query processing , the throughput of a query processing with a mapping arrangement MpC is the minimum value of a set of computation and communication throughputs. A congestion point is the place the throughput is settled, i.e. where the minimum value happens. On the off chance that the minimum value is TPcomp(Pr), the blockage is said to lie on the computation of Pr. For this situation, just task from Pr are considered to be moved to different partitions. This helps to reduce$\sum_{t\epsilon \mathrm{Pr}} w_r^t$ and what's more, along these lines increment TPcomp(Pr). Along these lines the throughput TP(MpC) then moves forward.

Likewise if TPcomm (Pr_i , Pr_j) is the minimum value, the congestion lies on the correspondence between segments Pr_i and Pr_j. For this situation, just move operations that decrease the correspondence weight between Pr_i to Pr_j are considered. Those are move operations including assignments which have a data association crosswise over Pr_i and Pr_j . Moving these assignments conceivably diminishes the correspondence weight amongst Pr_i and Pr_j and in this way potentially improve TPcomm (Pr_i , Pr_j).

The details of the CA algorithm are proven in below seudocode, labeled as steps 1 to 10. Taking queries G = (T,S) and a distributed platform H = (R,L) as inputs (step 1),CA starts by generating a randomly generated mapping configuration cur_map(step 2) before it gets into heuristic passes, CA stores the records of move operations in 3 lists moved_task, moved_par and moved_tp. every CA pass also begins with an empty history and a temporary mapping configuration tmp_map which is copied from the current mapping configuration cur_map (step 3). By means of evaluating the throughput system (Equation 5)[1] on tmp_map, the CA pass identifies the congestion factor at step 4. The type of congestion factor is used to determine the set of task T_{try} so that reallocating them can potentially improve the congestion point (steps 5, 6). If the congestion factor lies on the computation of Pr, T_{try} includes responsibilities in Pr. If the congestion factor lies on the communication of Pr_i and Pr_j , T_{try} consists of pairs of tasks, one in Pr_i and one in Pr_j ,that are connected via a data. In step 7, the algorithm examines move operations of re-allocating responsibilities in T_{try} that have not been moved all through the current pass, i.e. now not inside the listing moved_task. The result of this step is the move operation of task t to a every other partition P that brings the highest throughput compared to different examined move operations. If the move operation exists(at step 7 with yes), it is carried out to generate a new mapping configuration. The flow operation is also added to the history i.e. t is appended to moved_task ,p is appended to moved_par and new throughput value is added to moved_ tp. while all obligations were moved as soon as i.e. no move operation can be discovered i.e. step 7 and not using a. The list moved_tp is examined to discover maximum throughput value max_tp and its index k at step 8. Otherwise, the algorithm scans the records of move operations to find the maximum throughput cost max tp and its index k (step 8).The algorithm comes to a decision to update cur_map and continues a new CA pass if max_tp is better than the throughput of cur_map (step 9). If not, the algorithm terminates with cur_map as the output.

```
Algorithm 1: Congestion Avoidance Partitioning algorithm
Input: Queries $G = (T, S)$ , Distributed Platform $H = (R, L)$

Output: Mapping Configuration = $cur\_map$

1:set $cur\_map \leftarrow RndGen(G, H)$ ,

  set $cur\_tp \leftarrow TP(cur\_map)$

2: $tmp\_map \leftarrow cur\_map, moved\_task \leftarrow \theta$,

   $moved\_par \leftarrow \theta, moved\_tp \leftarrow \theta$

3: FIND congestion point $c$ of $tmp\_map$

4: If $c$ lies on computation of $P_r$ then set $T_{try} \leftarrow P_r$

5: Else set $T_{try} \leftarrow \{T_1, T_2 | T_1 \epsilon P_{r_i} \wedge T_2 \epsilon P_{r_j} \wedge \exists S_{T_1 T_2} \epsilon S\}$

6: FIND $T \epsilon T_{try}$ $moved\_task$ & $P \epsilon tmp\_map$ SUCH THAT
   $P \neq par(t)$ AND moving T to P gives the best throughput $T_p$
7: If T, P exist?
   then $moved\_task \leftarrow append(moved\_task, T)$,
   $moved\_par \leftarrow append(moved\_par, par(T))$ ,
   $moved\_tp \leftarrow append(moved\_tp, T_p)$
   $tmp\_map \leftarrow move(tmp\_map, T, P)$
   Go to step 3
8:  Else FIND index $k$ of the highest value in $moved\_tp$
    Set $max\_tp \leftarrow getElt(moved\_tp, k)$
9:  If $max\_tp > cur\_tp$
    Then set $cur\_tp \leftarrow max\_tp$
    For $i \leftarrow 1$ to $k$
    $T \leftarrow getElt(moved\_task, i)$
    $P \leftarrow getElt(moved\_par, i)$
    Set $cur\_map \leftarrow move(cur\_map, T, P)$
    Go to step 2
10: Else
     Mapping configuration= $cur\_map$.
```

4. Execution:

In this stage, divided query task executes severally on the distributed database system and got result from totally different nodes and merger node merge those succeeding data and provides required data to user . For this execution method Parallel execution algorithmic rule has used to execute query task at the same time on distributed network and computes minimum latency to improve throughput. Following algorithmic rule includes execution steps for parallel execution, It orders the relations in database to execute it with minimum latency.

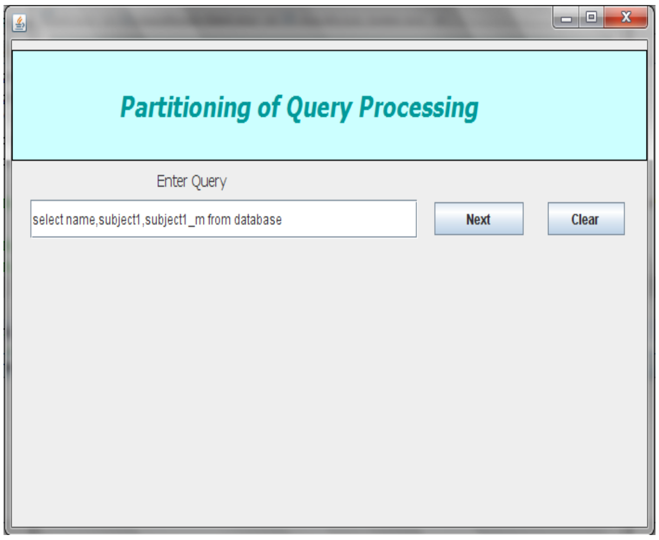
Algorithm 2: Query Execution (Parallel Algorithm):

Step 1: Order relations R_i such that $s_1 \leq s_2 \leq \cdots \leq s_m$.

Step 2: Consider each relation R_i in ascending order of size.

Step 3: For each relation R_j(j <i), construct a schedule to R_i that consists of the parallel transmission of the relation R, and all schedules of relations R_k(k <j). Select the schedule with minimum response time.

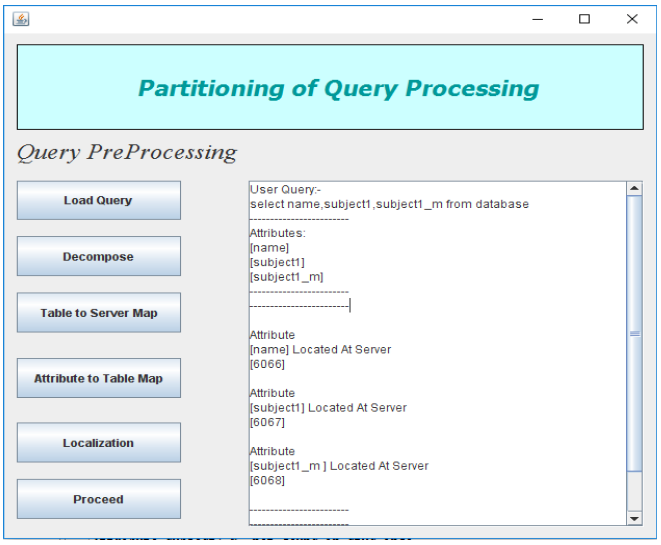
5. SNAPSHOTS



```
select name,subject1,subject1_m from database
```

Figure 2. Enter user query

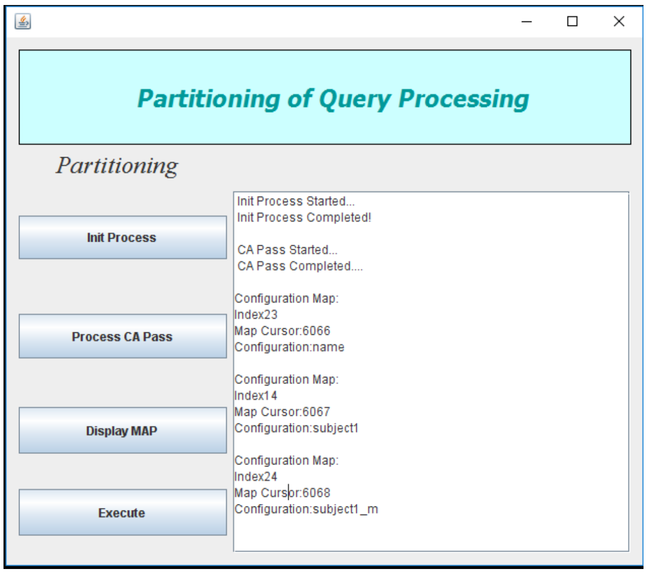
Figure 2 indicates the form that is helpful for user to enter any form of query (example select, insert, update etc) with its correct format of SQL to get data from totally different computers in distributed network.



```
User Query:-
select name,subject1,subject1_m from database
------------------------
Attributes:
[name]
[subject1]
[subject1_m]
------------------------
------------------------


Attribute
[name] Located At Server
[6066]

Attribute
[subject1] Located At Server
[6067]

Attribute
[subject1_m ] Located At Server
[6068]


------------------------
------------------------
```

Figure 3.Decomposition and Localization

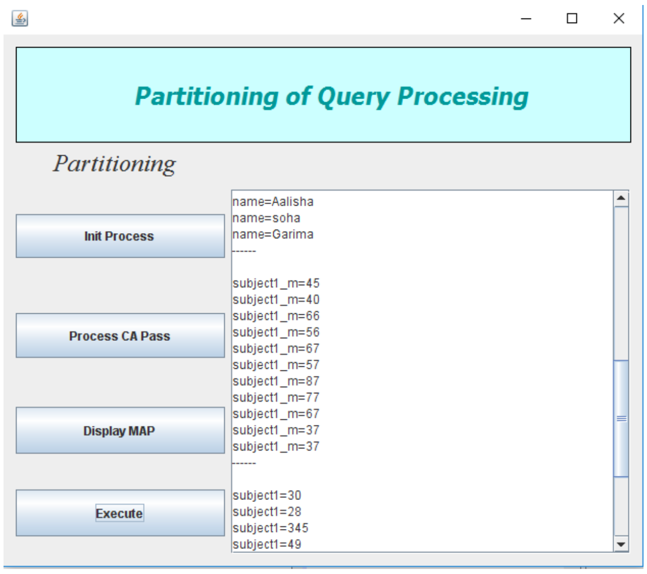
Figure 3 indicates the form for decomposition and localization of attributes inside the query. When user clicks on decompose button then it get output as fragmented or decomposed listing of attributes inside the query. For localize those attributes with its related server user clicks on localization button then it localizes attributes with servers and displays attributes and located server port no. as output.



```
Init Process Started...
Init Process Completed!

CA Pass Started...
CA Pass Completed....

Configuration Map:
Index23
Map Cursor:6066
Configuration:name

Configuration Map:
Index1 4
Map Cursor:6067
Configuration:subject1

Configuration Map:
Index24
Map Cursor:6068
Configuration:subject1_m
```

Figure.4 Partitioning of task

Figure 4 indicates the form of partitioning which performs the partitioning algorithm on task of query. When user clicks on init process button then it initializes the variables in algorithm and when user clicks on process CA pass button then it starts CA pass in algorithm to check congestion point in the system and when user clicks on display MAP button then it gets output of CA i.e. mapping configuration of these attributes in query.



```
name=Aalisha
name=soha
name=Garima
------


subject1_m=45
subject1_m=40
subject1_m=66
subject1_m=56
subject1_m=67
subject1_m=57
subject1_m=87
subject1_m=77
subject1_m=67
subject1_m=37
subject1_m=37
------


subject1=30
subject1=28
subject1=345
subject1=49
```

Figure 5. Execution of query

Figure 5 indicates the form for execution of query task. When user clicks on execute button then it gets output as values of attributes for select query and for another query (insert, update, delete etc.) it updates database on related host.

6. RESULT ANALYSIS

In this section we evaluate the performance and memory requirement by comparing our implemented system with another system where partitioning algorithm has not implemented (without partitioning).

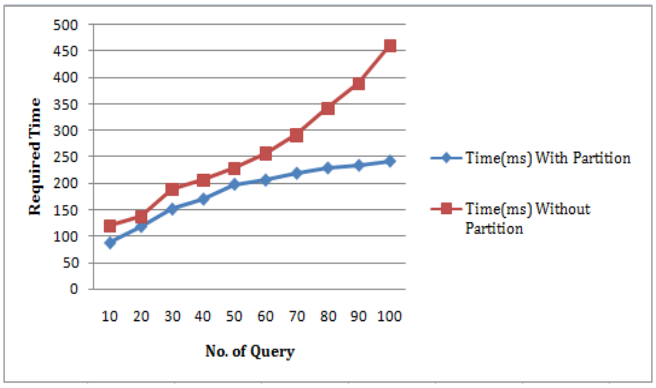


Chart-1: Time comparison of system with partitioning and without partitioning

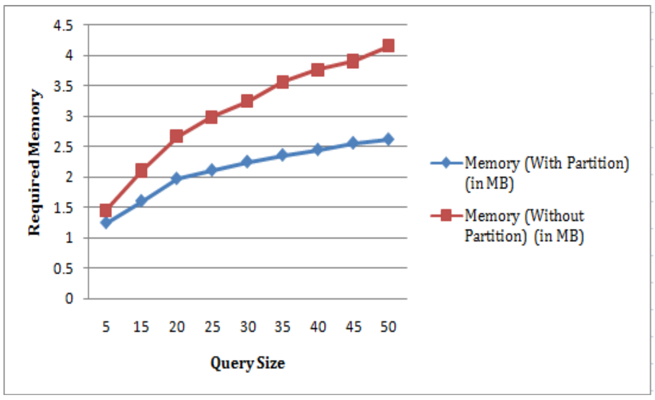


Chart-2: Memory size comparison of system with partitioning and without partitioning

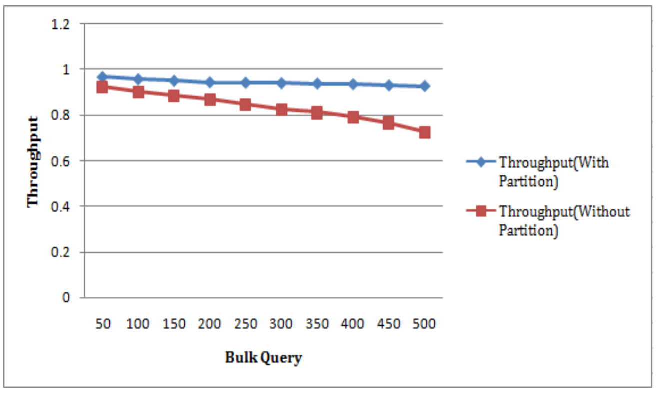


Chart-3: Throughput comparison of system with partitioning and without partitioning

In chart-1 we compare the time required for query processing by using partitioning algorithm with another system which not using partitioning algorithm for no. of queries from 10-100 in distributed network. Similarly in chart-2 we compare memory required for query processing with partitioning and without partitioning algorithm.

After comparing time and memory requirements of system, in chart-3 we compare the overall throughput of the system of query processing with partitioning and system without partitioning algorithm.

7. CONCLUSION

We have made survey of different literatures and studied different techniques for graph partitioning and query processing. To gain better throughput of query processing many optimization methods are used and many techniques are described for graph partitioning to divide workload of PEs.

In this paper, we used graph partitioning methods to partition the workload of query processing to optimize throughput on distributed environment. Query preprocessing method used for fragment and localize input query and partitioning algorithm applied on fragmented query task to gain better throughput which has been proved in section 6 by comparing system with another system where partitioning algorithm has not used.

8. FUTURE WORK

There is much scope for improve concurrency while accessing database through multithreaded algorithms. Improve disinterestedness and redundancy so that data availability will get increased. Need to design data sharing protocol over application layer to improve data sharing and representation between two nodes.

ACKNOWLEDGEMENT

There have been many contributors for this to take shape and authors are thankful to each of them.

REFERENCES

[1] Vu Thien Nga Nguyen , Raimund Kirner ."Throughput-driven Partitioning of Stream Programs on Heterogeneous Distributed System", IEEE Transactions on Parallel and Distributed Systems,2015.

[2] R. Khandekar, S. Rao, and U. Vazirani. "Graph partitioning using single commodity flows". In Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing, STOC '06, pages 385–390, New York, NY, USA, 2006.

[3]B. L. Chamberlain. "Graph partitioning algorithms for distributing workloads of parallel computations", Technical report, 1998.

[4] Fan Yuanyuan, Mi Xifeng "Distributed Database System Query Optimization Algorithm Research",978-1-4244-5539-3/2010 IEEE.

[5] Kunal Jamsutkar "Query Processing Strategies in Distributed Database" Journal of Engineering, Computers

& Applied Sciences (JEC&AS) ISSN No: 2319-5606 volume 2, No.7, July 2013

[6]B.M. Monjurul Alom, Frans Henskens *Query Processing and Optimization in Distributed Database Systems* .IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.9, September 2009

[7] R. Hevner and S. B. Yao, "Query Processing in distributed database systems," IEEE Trans. Software Eng., vol. SE-5, pp. 177-187,May 1979.

[8] Donnald Kossman."The State of the Art in Distributed Query Processing" ,*University of Passau,* ACM Computing Surveys, Vol. 32, No. 4, December 2000, pp. 422–469.

[9] C.Walshaw and M. Cross. *Mesh partitioning:"*A multilevel balancing and refinement algorithm", SIAM J. Sci. Comput., 22(1):63–80, Jan. 2000.

[10] W. Chen. "Task partitioning and mapping algorithms for multicore packet processing systems", Master's thesis, University of Massachusettes - Amherst, Massachusettes, USA, 2009.

[11] L. Hagen and A. B. Kahng. "New spectral methods for ratio cut partitioning and clustering", Trans. Comp.-Aided Des. Integ. Cir. Sys.,11(9):1074–1085, Nov. 2006.