ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Optimized Multi Data Aggregators to Support High Web Traffic Push Servers

[1] MR N . HARI KRISHNA MCA. [MTECH], [2] MR MD . SIRAJUDDIN . MTECH

1FINAL M TECH STUDENT, 2ASSIATNANT PROFESSOR,

[1,2]DEPT OF COMPUTER SCIENCE AND ENGINEERING, SRI MITTAPALLI COLLEGE OF ENGINEERING, TUMMALAPALEM, GUNTUR (DT)

Abstract: Continuous queries are persistent queries that can transform a passive web into an active environment by providing time varying dynamic query results useful for online decision making. For scalable handling of push based data dissemination, prior approaches used a network of data aggregators. Their implementation required Greedy Heuristics Algorithm along with pre configured incoherency bounds to manage both multiple aggregators and multiple clients for supporting server push based communications. Existing heuristic-based approaches can only explore a limited solution space and hence may lead to sub-optimal solutions. So we propose to use an adaptive and cost-based approach. In a network of data aggregators, each dedicated and judiciously chosen aggregator serves a set of data items at specific coherencies. It involves decomposing a client query into sub-queries and executing sub-queries using aggregators with their individual sub-query incoherency bounds. Our cost model takes into account both the processing cost and the communication cost unlike prior approaches. Adaptive and cost-based approach has better performance in terms of both processing and communication cost than plain Greedy Heuristics approach and a practical implementation validates the proposed claim.

Index Terms: Continuous Queries, Distributed Query Processing, Data Incoherence, Network Data Aggregator.

I. INTRODUCTION

Distributed stream processing has been main commercial gaining process in research attentions in the recent years. In such a system, queries submitted by the clients (e.g., continuous queries monitoring the streams or ad hoc queries on the historical and current status) would be distributed to the various processing servers for processing. To evaluate the client queries, the streaming data have to be disseminated from the sources to the distributed servers. Due to the continuity and massiveness of the data, it is critical and challenging to design an effective, efficient and scalable dissemination system.

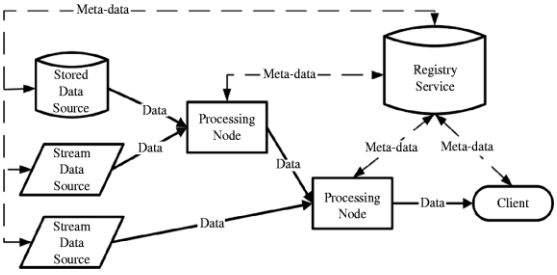
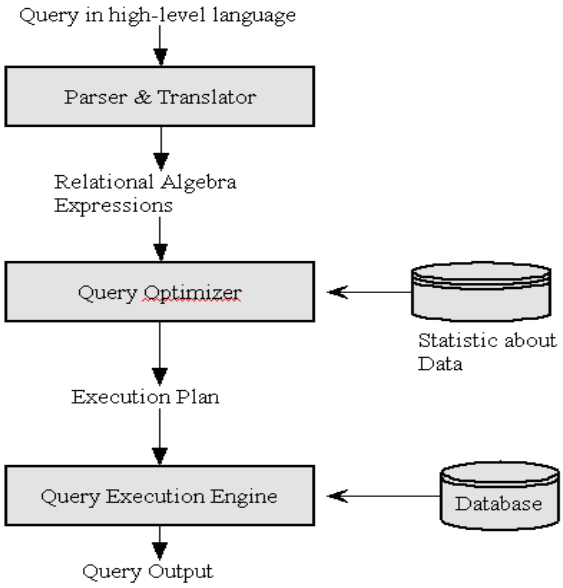


Figure 1: Distributed query processing.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

As shown in the above figure every user data can be stored in source data, and then every user can access that data using processing node. By using processing node users can register their data in register service. In this process every client connects with equivalent server according their requirements. Client enters into communication systems then clients submit queries to the servers with their own preferences on data coherency requirements. Data coherency is an affective design for data processing between different clients. Based on the client requirements of the running queries, each server would have its own interesting object set as well as its coherency requirement of each interesting data object.



According to the client requirement servers are organized into one or more dissemination trees (with the data source being the root node) so that data/messages are transmitted to each server through its ancestors in the dissemination tree. Each node present in tree would selectively disseminate only interesting client data to its child nodes by filtering, it includes only necessary data, and it eliminates unnecessary data. Consider the above process present between client and server communication. Traditionally greedy heuristic algorithm for accessing services from server with their requirements. In that multiple aggregator's support multiple clients for supporting server push based services. But these results are for performing limited services. In this paper we are proposing a Adaptive Cost based approach, in that each dedicated judiciously chosen aggregators a set of items at specific coherences. Compared to existing greedy heuristic algorithm our proposed work will give efficient results.

II. RELATED WORK

We first formulate the problem by presenting the system model, the definition of the metric as well as the formal problem statement. Then we motivate our work by identifying the potential problems of the existing techniques.

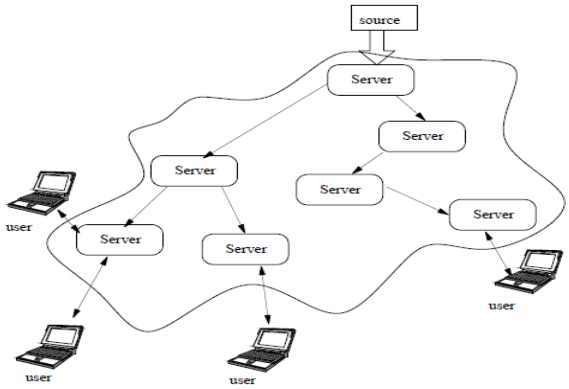


Figure 2: Query formulation regarding client request.

Every client submits requests according their relational requirement involving a subset of data objects through a server (or the data source), and specifies a preference on the coherency on each data

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

object. The possible metrics are number of changes since the updates accessed with each client. These results can be obtained using greedy heuristic model approach. But these results are not accessed in large data storage of clients. So finally we are composed Adaptive Cost based `approach for large data storage client requests.

III. EXISTING APPROACH

Continuous queries are persistent queries that allow users to receive new results when they become available. While continuous query systems can transform a passive web into an active environment, they need to be able to support millions of queries due to the scale of the Internet. Continuous queries allow users to obtain new results from a database without having to issue the same query repeatedly. In order to handle a large number of users with diverse interests, a continuous query system must be capable of supporting a large number of triggers expressed as complex queries against resident data storages. Push, or server push, describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server. It is contrasted with pull, where the request for the transmission of information is initiated by the receiver or client. Uses server push based techniques for initiating communications. Push services are often based on information preferences expressed in advance. This is called a publish/subscribe model. A client might "subscribe" to various information "channels". Whenever new content is available on one of those channels, the server would push that information out to the user.

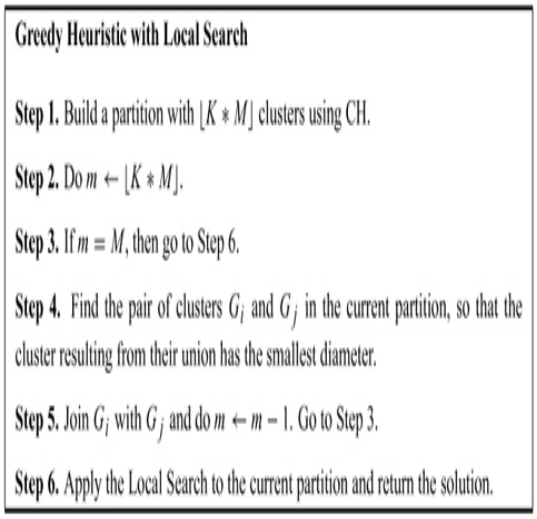


Figure 3: Greedy Heuristic Local search algorithm

For scalable handling of push based data dissemination, we use a network of data aggregators. Data refreshes occur from data sources to the clients through one or more data aggregators. *Heuristic* refers to experience-based techniques for problem solving, learning, and discovery. Uses Greedy Heuristics Algorithm along with pre configured incoherency bounds to manage both multiple aggregators and multiple clients thus delivering a better performance.

IV. PROPOSED APPROACH

Prior Approaches use Greedy Heuristics Algorithm along with pre configured incoherency bounds to manage both multiple aggregators and multiple clients for supporting server push based communications. Query optimization strategies developed using Greedy Heuristics Algorithm depends on processing cost only. Existing heuristic-based approaches can only explore a limited solution space and hence may lead to sub-optimal solutions.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

So we propose to use an adaptive and cost-based approach. Our cost model takes into account both the processing cost and the communication cost. Adaptive and cost-based approach has better performance in terms of both processing and communication cost than plain Greedy Heuristics approach.

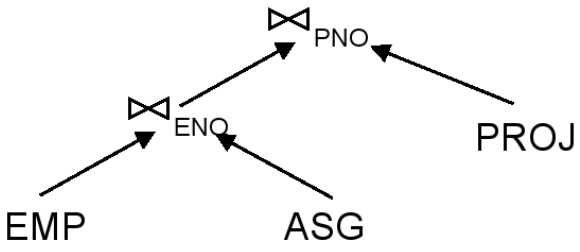
V. ADAPTIVE COST BASED APPROACH

The adaptation cost approach operates in rounds. We are representing in the form of tree notations. The root node initiates each round by creating a token. Only when a node holds a token, could it make an adaptation attempt. In this cost based approach we are performing following three categories of data.

Query Optimization: Process of producing an optimal (close to optimal) query execution plan which represents an execution strategy for the query.

Example: 3 equivalent query trees (join trees) of the joins in the following query

Select Ename, Resp
From Emp, Asg, Proj
Where Emp.Eno=Asg.Eno **And** Asg.Eno=Proj.Pno.



Optimistic Query results.
Search for optimal solutions around a particular starting point.

Cost Model: There are two types of cost models present in data aggregation. Firstly we are reducing total time represented in each cost component and then we are verifying response time according to the client request.

Response time: Elapsed time between the initiation and the completion of a query

Response time =TCPU * #seq instructions + TI /O *#seq I /Os +TMSG * #seq messages + TTR * #seq bytes

-where #seq x (x in instructions, I/O, messages, bytes) is the maximum number of x which must be done sequentially.

Total time: Sum of the time of all individual components. Local processing time: CPU time + I/O time. Communication time: fixed time to initiate a message + time to transmit the data

Total time =TCPU _ #instructions + TI /O * #I /Os + TMSG * #messages + TTR *#bytes.

Algorithm: Adaptive Attempt

Begin:

MaxBene fit—0; t—NULL

For each possible transformations t1involving the children and grand children do

{

If (maxBenefit < benefit (t1) then

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Max Bene fit -----Bene fit (t1)

t----(t1)

}

If(t!= NULL then perform t;

For each child n_j do

If (n_j) is not a leave node them

Send one copy of the token to n_j

End

The above diagram presents the operations to be executed in a node that receives a token. Each node receives a token can make its own decision independently without any synchronization with the other nodes. Instead of allowing every node to try all kinds of transformations, we restrict each node to consider only the transformations involving its children and grandchildren. These include promoting a grandchild (node promotion), demoting a child (node demotion), swapping a child and a grandchild (parent-child swap and uncle-nephew swap), swapping two grandchildren (cousin swapping), and moving a grandchild from one child to another child (nephew adoption).

VI. EXPERIMENTAL RESULTS

We set the average values of these times as 5ms and 1ms respectively (which may vary in our experiments), and set the minimum values as 1ms and 0.125ms respectively. The source server's expected filtering time and transmission time are always set to the minimum value to model an enterprise class server.

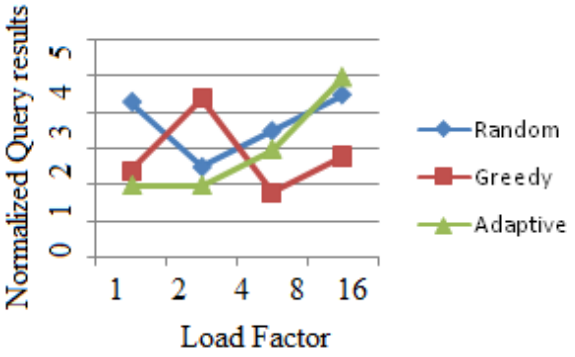
Algorithm : Process Message

```
1  begin
2      while true do
3          wait for a new message msg;
4          HandleMsg(msg);
5          if state = IDLE||WAIT then
6              for each wait ∈ Q_ready do
7                  wait ← Q_ready.Dequeue();
8                  PerformAdapt(wait.tree);
9                  send a token message to each node in
                   Child[wait.tree];
10                 continue;
11             for each msg in Q_token do
12                 remove msg from Q_token;
13                 HandleMsg(msg);
14             for each msg in Q_hold do
15                 remove msg from Q_hold;
16                 HandleMsg(msg);
17                 if state = HOLD then break;
18 end
```

Given the expected filtering time t p i and transmission time tc i for a node, the exact filtering time and transmission time of each message are drawn from two respective exponential random variable with expected values as t p i and tc i respectively.

We study the cost of performing adaptations using our C implementation. To examine the cost of making adaptation decisions, we use a node that serves 100 objects and try estimating 100 possible decisions. We found that td k _ 0:6ms for both the single-tree and multi-tree approaches. To keep the adaptation cost affordable, we have to set an appropriate adaptation period Td. For example, if we can afford 5% of the CPU time for adaptation, we can set the adaptation period of this testing node.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)



For example, if this node serves 10,000 objects, we have to set the adaptation period larger than or equal to 12 seconds. Therefore, to keep the adaptation responsive, the number of objects served by each server and the number of children and grandchildren should be kept to a certain limit.

VII. CONCLUSION

For supporting continuous queries for the users, the service provider is maintained and managed by a single resource builder. In order to handle a large number of users with diverse interests, a continuous query system must be capable of supporting a large number of triggers expressed as complex queries against resident data storages. Push services are often based on information preferences expressed in advance. This is called a publish/subscribe model. For scalable handling of push based data dissemination, we use a network of data aggregators. So we propose to use an adaptive and cost-based approach. Our cost model takes into account both the processing cost and the communication cost. Adaptive and cost-based approach has better performance in terms of both processing and communication cost than plain Greedy Heuristics approach.

VIII. REFERENCES

[1] Rajeev Gupta, Kirthi Ramaritham, " Query Planning For Continuous Aggregation Queries Over A Network Of data Aggregators" vol 24, No. 6. June 2012.

[2] S.Shah, K. Ramamritham, and P. Shenoy, "Maintainin Coherency of Dynamic Data in Cooperating Repositories," Proc. 28th Int'l Conf.Very Large Data Bases (VLDB), 2002

[3] Y.Zhou, B. Chin Ooi, and K.-L. Tan, "Disseminating Streaming Data in a Dynamic Environment: An Adaptive and Cost Based Approach," The Int'l J. Very Large Data Bases, vol. 17, pp. 1465-1483, 2008.

[4] S. Agrawal, K. Ramamritham, and S. Shah, "Construction of aTemporal Coherency Preserving Dynamic Data Dissemination Network," Proc. IEEE 25th Int'l Real-Time Systems Symp. (RTSS), 2004.

[5] R. Gupta, A. Puri, and K. Ramamritham, "Executing Incoherency Bounded Continuous Queries at Web Data Aggregators," Proc. 14th Int'l Conf. World Wide Web (WWW), 2005.

[6] C.Olston, J. Jiang, and J. Widom, "Adaptive Filter for Continuous Queries over Distributed Data Streams," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2003.

[7]R.Gupta and K.Ramamritham,"Optimised Query Plary planning of continuous Aggregation Queries in Dynamic Data Dissemination Networks",WWW.2007.

[8] Yongluan Zhou _ Beng Chin Ooi _ Kian-Lee Tan," Disseminating Streaming Data in a Dynamic Environment: an Adaptive and Cost-Based Approach", School of Computing, National University of Singapore, Singapore. E-mail: fooibc,tanklg@comp.nus.edu.sg