Relational Algebra Coarse Grained Query Cost Models for DDFDs

Paul D Stone, Patrick Dantressangle, Graham Bent, Abbe Mowshowitz, Andi Toce, Boleslaw K Szymanski

Abstract— **This paper defines a number of Query models used to visualize and explore the performance of Relational Algebra queries evaluated in a Dynamic, Distributed, Federated Database with a number of distinct network topologies. Query costs are modelled at a coarse grained level, using a small number of parameters and formulating only the dominant or average behaviours of the queries and topologies considered. This allows us to determine the prevailing factors which impact query performance, and provide a framework to refine and focus on more specific behaviours.**

Index Terms—**Database, Distributed, Federated, GaianDB, Network Topology, Query Optimisation, Relational Algebra.**

I. Introduction

THE Gaian Database [1] is a dynamic, distributed federated database which combines the principles of large distributed databases, database federation, and network topology in a dynamic, ad-hoc environment.

The GaianDB has been shown to be scalable for simple queries [2] but can be enhanced to optimise complex queries such as joins, aggregate functions and nested queries. In addition to identifying optimal query plans, we can determine how to establish a network overlay topology to allow optimal performance of queries.

It is recognised that different query strategies and different network topologies are optimal in different situations (such as the number of nodes in the network or the size or distribution of the data to be retrieved).

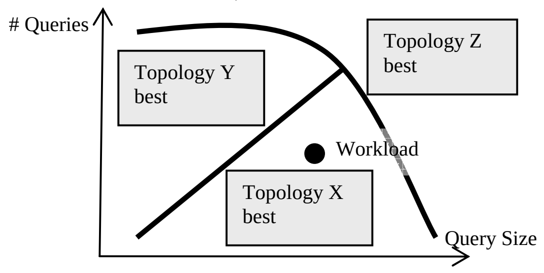

Fig. 1 Optimal network topology varies for different query workloads

A useful step in formulating query optimisation algorithms is to understand the main dimensions of variance and to model query costs using coarse averages at a high level to understand where different topologies and query strategies become optimal at extremes of these variable dimensions. Figure 1 illustrates the principle that different network topologies can result in the lowest query cost for different query workloads.

This paper presents the first steps to produce these coarse grained query cost models. This paper has so far considered simple select queries for Preferential Attachment topology and a Content Addressable Network based on a Hypercube.

II. Network Topology Models

The networks must be considered to be dynamic; nodes will frequently join and leave the network and must be able to reconnect to more appropriate nodes as the utility of different nodes changes.

To form robust networks in a scalable way, the decision to which existing nodes a new node connects should be made on a peer-to-peer basis without need for centralised management.

A. *Hypercube Topology in a Content Addressable Network*

In a network of nodes, we can define a hypercube topology by assigning a hypercube label to each node. The imposition of a hypercube structure should provide advantages when routing queries to known destinations. Hypercube labelling also provides an efficient mechanism to broadcast messages, where we can guarantee that a message will be received by each node once and only once in the broadcast procedure.

Reference [4] outlines a method for constructing and maintaining a Content Addressable Network (CAN) within a peer-to-peer network of labelled nodes. This provides the ability to determine where specific content exists within a network and to send a query to only these nodes of interest.

Instead of broadcasting a query to all nodes of a distributed database, we propose to keep details of which nodes have which Logical Tables, so a query can determine which nodes host a fragment of the table and target the query only to those nodes.

We propose the use of a CAN to maintain a Hypercube labelled network, and the Logical Tables contained therein.

B. *GaianDB Preferential Attachment Topology*

The GaianDB implements a preferential attachment algorithm so a joining node is more likely to connect to a node which already has a large number of connections. The aim of this approach is to reduce the diameter of the network. The number of hops between nodes is reduced to minimise the total time required to perform the broadcast-style queries of the prototype Gaian Database.

Each node that joins a GaianDB network broadcasts a request for connections and existing, networked nodes respond with a delay that depends on the number of connections that it already has; the more connections a node has, the shorter the delay will be. The joining node will accept the first connections it receives, so considering the delay, the node is more likely to connect to a node which already has many connections.

The average path length for this topology was initially calculated using [3], but this was a significant underestimate due to assumptions that the network is infinite size. By analyzing empirical topology data, we discover that the average path length is close to $\log_2 N/2$, which is the same as for a hypercube.

Broadcasting a Gaian Query to all nodes results in 2N messages being sent. The GaianDB topology has a fixed average vertex degree, due to each node forming 2 outgoing connections to other nodes. In other random graphs, vertex degree may vary with network size.

III. Select Query Cost Models

This section shows the cost formulae used. We consider the cost of a query in terms on the total network bandwidth required to evaluate the query.

The query costs shown here are coarse-grained averages derived from a small set of parameters such as the number of nodes and the data density, resulting in a coarse-grained assessment of query costs as shown in the following sections. The costs formulae will be refined with further research.

A. Query Cost Formulae

N is the number of nodes in the network,
LT_x = proportion of fragments of the Logical table named X,
PL_N = Average path length in a network with N nodes
SLL = Size of a logical table lookup message and response (per network step)
SQ = Size of a query message and standard (no data) response,
SQR = size of data results per logical table fragment,

Hypercube topology with Content Addressable Network:

The cost of logical table lookup in a CAN is $PL_N * SLL$

The cost of sending the query to the specific locations with that logical table is $N * LT_x * PL_N * SQ$

The cost of retrieving results is $PL_N * SQR * N * LTx$

$$Cost(Select, hypercube\ CAN) = PL_N * SLL + N * LT_x * PL_N * SQ + PL_N * N * LT_x * SQR$$

Gaian topology:

The cost of sending the query to all nodes is $2 * N * SQ$

The cost of retrieving results is $PL_N * N * LT_x * SQR$

$$Cost(Select, Gaian) = 2 * N * SQ + PL_N * SQR * N * LT_x$$

B. Topology Comparison

We have taken representative values of the specified variables and then vary them to see which topologies have the lowest cost using these cost formulae.

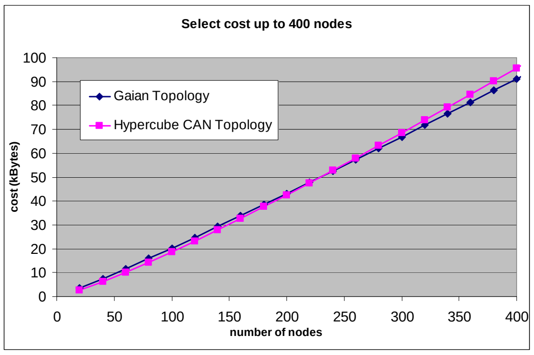

Fig. 2 Select cost, varying the number of nodes

Fig. 2 shows that in networks with less than 250 nodes, the hypercube topology has a lower cost that the GaianDB topology. In networks with more than 250 nodes, the GaianDB topology has a lower cost.

Fig. 3 shows that when logical table data is present on less than 4 in 10 nodes ($LT_x = 0.4$) then a hypercube topology is cheapest, due to being able to target a small subset of the total number of nodes. When LT_x is more than 0.4, the GaianDB query is cheapest, as the hypercube CAN imposes an overhead of sending a separate message to each node.

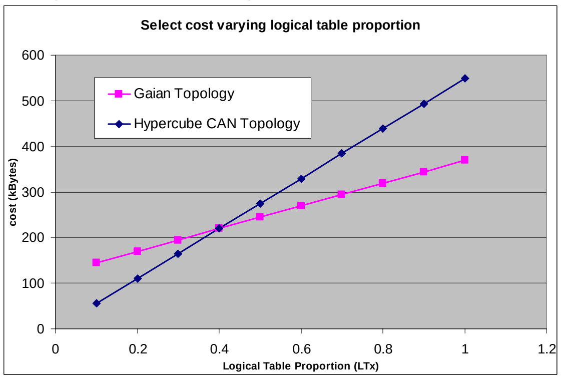

Fig. 3 Select cost, varying the proportion of logical tables

IV. Conclusions

These initial results suggest that networks with more nodes favour a random preferential attachment topology (i.e. GaianDB), but where a Logical Table is sparsely present in the network, a Hypercube CAN topology is cheaper.

We intend to expand and validate these Coarse Grained query cost model to include join and aggregate queries, alternative network topologies, such as that identified in [5] and to include the cost of network maintenance.

V. Acknowledgement

Research was sponsored by US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon

References

[1] G Bent, P Dantressangle, D Vyvyan, A Mowshowitz, V Mitsou, "A Dynamic Distributed Federated Database", "*Proc. 2nd Ann. Conf. International Technology Alliance*" 2008.
[2] G. Bent, P. Dantressangle, P. Stone, D. Vyvyan, A. Mowshowitz, "Experimental Evaluation of the Performance and Scalability of a Dynamic Distributed Federated Database", "*Proc. 3rd Ann. Conf. International Technology Alliance*" 2009.
[3] M. Newman, S Strogratz, D Watts "Random graphs with arbitrary degree distributions and their applications" Physical Review E, Volume 64, 2001.
[4] P. Fraigniaud, P. Gauron, "D2B: a de Bruijn Based Content-Addressable Network", 2006.
[5] Z Wang, E Bulut, B K Szymanski, G Bent, A Mowshowitz, P Stone, "An Energy Efficient Dynamic Location Server Hierarchy for Mobile Ad Hoc Networks", 2010