Query Processing in Highly Distributed Environments*

Akira Kawaguchi[1], Nguyen Viet Ha[2], Masato Tsuru[2], Abbe Mowshowitz[1], and Masahiro Shibata[2]

Abstract This paper will demonstrate a novel method for consolidating data in an engineered hypercube network for the purpose of optimizing query processing. Query processing typically calls for merging data collected from a small subset of server nodes in a network. This poses the problem of managing efficiently the exchange of data between processing nodes to complete some relational data operation. The method developed here is designed to minimize data transfer, measured as the product of data quantity and network distance, by delegating the processing to a node that is relatively central to the subset. A hypercube not only supports simple computation of network distance between nodes, but also allows for identifying a node to serve as the center for any data consolidation operations. We will show how the consolidation process can be performed by selecting a subgraph of a complex network to simplify the selection of a central node and thus facilitate the computations required. We will also show a prototype implementation of a hypercube using Software-Defined Networking to support query optimization in a distributed heterogeneous database system, making use of network distance information and data quantity.

1 Introduction

Today, scientists and engineers are building a unique computational infrastructure and testbed to support large-scale computing and information management that encompasses big data, data science, data analytics and visualization research. The instruments and infrastructure will be used for developing next-generation algorithms and software platforms to support these efforts while allowing precise experimentation with multiple architectural options. As we attempt to solve increasingly

[1] Department of Computer Science, City College of New York, NY 10031, USA e-mail: {akawaguchi, amowshowitz}@ccny.cuny.edu

[2] Department of Computer Science and Networks, Kyushu Institute of Technology, Iizuka, Fukuoka, 820-8502 Japan e-mail: {ha, tsuru, shibata}@cse.kyutech.ac.jp

* This material is based upon work supported by the National Science Foundation under Grant No. 1818884.

complex problems, a combination of computing platforms through effective data networking is often desirable. A wide range of computing platforms, from traditional clusters or cloud servers can be connected through the fast, high bandwidth networks, including 5G. The landscape of network services is becoming increasingly diverse with recent advances in technology, particularly applications of visual computing and machine learning. This new technology can make dramatic improvements in the management of large urban environments. In edge computing [1, 2], geographically distributed applications can run throughout the network utilizing the computing paradigm that relies on user or near-user (network edge) devices to conduct required processing. This architecture extends cloud computing with more flexibility than that found in conventional networks by allowing the integration of a massive number of components and services [35, 34, 30].

The challenges of information management derive in part from the dynamic and distributed features of operations in such a network. In particular, to exploit fully computing at the edge it is necessary to take account of the effect of querying on message traffic in the network. For instance, consider a smart city [18, 10] covered by e-services and e-resource management using a collection of sensors, communication devices, and data-processing facilities. The absence of centralized management dictates policies designed to maximize the use of edge nodes which typically have relatively modest storage and processing power. Edge nodes must thus work together in information processing tasks, and this occasions the movement of data between nodes. To minimize the message traffic, innovative strategies are required. One such strategy detailed here is the development of an engineered overlay network structured as a hypercube graph [32, 33, 23]. This strategy was originally proposed in the International Technology Alliance project [19, 3, 20, 21]. Research undertaken in that project partially demonstrated the feasibility of using hypercube routing for query optimization in a Dynamic Distributed Federated Database (the GAIAN DB) built by IBM-UK [4]. The choice of hypercube allows for determining the number of inter-node hops by computing the Hamming distance between corresponding node labels. This leads naturally to reexamination of distributed query optimization with a view to minimizing the total network traffic associated with querying.

Assuming the presence of distributed databases connected by a hypercube network, our simulation study [16] has confirmed a significant theoretical advantage of a query optimization approach that incorporates inter-node distances into the cost model. The query plan applied by this cost model would produce much better plans in terms of the overall network usage for data transmission. Our study also confirmed a significant advantage of this cost model in the “peer-to-peer collaboration” of databases in the network, and compared performance of the greedy algorithm to that of dynamic programming. The latter gives useful results in the case of a small number of participating nodes, but computational complexity limits its usefulness for a large number of nodes. Based on these findings, we will apply practical considerations to develop another approach, a so called “delegation to centrality” model, to reduce data transmission cost in a scalable way for performing database operations in the network with a larger number of nodes. A hybrid approach that combines peer-to-peer collaboration and delegation to centrality is also feasible. This approach exploits a divide-and-conquer principle such as applying peer-to-peer collaboration for small clusters and then gathering results by utilizing delegation to centrality or vice verse. Therefore, we must take account of the performance differences and trade-offs of both approaches. Fine tuning the optimization based on advanced techniques covered in [8, 15, 31] can be introduced later. A comparative performance analysis needs a systems environment with a typical distributed database application. To this end, we have developed a relational-oriented but heterogeneous database application that runs on a hypercube generated by the Software Defined Networking. In this paper, we report on these studies to showcase the engineered hypercube as an efficient and practical distributed database tool, and also to provide preliminary performance data obtained from an experiment.

2 Approach for Distributed Query Optimization

The engineered hypercube allows for determining inter-node distances cheaply, but query performance is dependent on the number of participating nodes, and the sequence of operations required [7, 16, 28]. A key element in our proposed research is the exploitation of knowledge of the network structure to facilitate optimization. To characterize this computing situation formally, let us consider a distributed relational database environment. We can use it to provide consolidated cost estimates for logical tables, which are in effect fragmented relations, thus decreasing the query-execution-plan search space. Most importantly, we consider internode network distances and the role of topology. Since many nodes in a network may rely on batteries and wireless radios, we should like to minimize network information transfer to maximize network availability and lifetime.

2.1 Problem Definition

How to distribute the date over the nodes in a network could affect the relative advantage of query optimization. This is especially true when the data is widely spread over the network. The optimization problem involves finding the most cost efficient way of performing the actions specified in a query. For the distributed processing environment, the optimization must take account of the amount of data exchanged between nodes over the network. First of all, let us define a query Q to be expressed in the form

$$Q(R_1,\ldots,R_m) = R_1 \circ R_2 \circ \cdots \circ R_m$$

where R_i ($1 \leq i \leq m$ with $m > 2$) are relations in a relational database, and the operator $\circ$ is a set-oriented operation, either union ($\bigcup$), intersection ($\bigcap$), subtraction ($-$), or join ($\bowtie$). The query is expressed in standard relational algebra notation — parentheses may be added to specify explicitly the order of operations. To start with the most fundamental optimization, we do not nail down the details of select and project operations, nor include aggregate operations for Q such as computing an average and sum of values which will be applied after consolidating all the data in some way. Furthermore, we do not handle a trivial query so as to dispatch Q to each database and gather its response to construct a final result. We are most interested in dealing with a complex decision support or analytic query that mandates shuffling R_1 through R_m over the network. To shed light on such an application, consider the following examples.

Example 1. Suppose there is a criminal investigation involving sophisticated drug smuggling into country A from country B. Furthermore, suppose that international law enforcement can access a relation R_1 holding records of recent immigrants from A, and a relation R_2 holding records of recent immigrants from B. The country A owns R_1 while the country B owns R_2. Suppose also that law enforcement has obtained records of convictions of their spouses, as a relation R_3. A relation R_4 holds records of those who filed customs declarations at country A, and a relation R_5 holds records of those who filed customs declarations at country B. Relations R_4 and R_5 are owned by A and B, respectively. Law enforcement could apply a query $Q(R_1,\ldots,R_5) = (R_1 \cap R_2) \bowtie R_3 - (R_4 \cup R_5)$ to identify individuals suspected of being drug smugglers and thereby prevent criminal acts or to take preventive measures to enhance security at immigration. The distributed query Q enables the two countries to obtain information without constructing a shared, centralized database.

Note that the relation instances could be concrete instances or temporary materializations resulting from some localized operations at the hosting nodes. Note also that the schemas of those that appear in the operands of union, intersection, and subtraction are assumed union-compatible (i.e., having identical structure). These presuppose availability of a spectrum of optimization methodologies, especially for select and project relational operations. For instance, selection of date and time windows as well as age ranges for R_1 and R_2 can be applied for reducing data space prior to data gathering and consolidation. A sophisticated query optimizer may be able to parse the query and rewrite it to apply predicate conditions applicable for data selection at each site, thereby reducing the amount of data to exchange for the final computation.

2.2 Communication-cost in Distributed Query Optimization

For a typical query the total number n of nodes in the network is much larger than the number m of the data-hosting nodes. The challenge is to find an optimal distributed execution plan for this m-fold set query such that the plan guarantees the smallest total amount of data transmission over the n-node network. Specifically, our approach is to investigate the next two types of optimization paradigm:

Peer-to-peer collaboration: a pair of nodes hosting R_i and R_j will communicate and send the instance, either from R_i to R_j or vice verse. The sent node will compute $R_i \circ R_j$ and communicate again to combine its result with a partial result held by another node in the party of data-hosting nodes. The peer-to-peer collaboration will continue until it derives the final result. Deriving the execution plan relies on the estimate of the result size at each $\circ$ operation; the estimate cascades to approximate the size of the next operation. The plan may improve by reflecting the actual result size, but this may be at the expense significant computational overhead.

Delegation to centrality: a group of m-nodes hosting relations will send their data instances to a so-called central node that can be reached by the smallest total distance from the group. The central node will then execute the query using local optimization for those gathered instances and hold the result. The central node could be one of the group or a certain node that does not host any relation at all but has the power to process the query. Similarly, the central node is not necessarily the node that generated the query. The algorithm will select the central node to minimize the overall data transfer with the data server nodes at query processing.

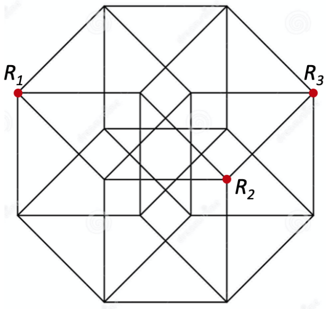


Fig. 1 Query process on hypercube

The heart of distributed query optimization is to minimize the overall amount of data transmission on the network as data has to move over the network to perform operations. Classic query optimization assumes that there is no cost associated with obtaining knowledge of internode distances or the overhead of data transmission. One critical measure of message traffic is the amount of data to be moved multiplied by the distance moved [16, 17]. Thus, message traffic associated with querying can be reduced by minimizing the product of data size and distance of travel as a network hop count at each operation involving several participating nodes in the execution of a query. Our previous study of [22, 16] showed that better plans could be found with the use of internode distances than without. The reasoning here is to minimize the bandwidth usage of the entire network system. Reducing the total occupancy of data in the network will mitigate network congestion and yield faster query response time; the longer the distance to send data, the higher the occupancy of data in the network and thus the slower the query response time. This is especially true when the network accommodates a large number of queries requiring data from different nodes.

Example 2. Consider the processing of a query $Q(R_1,R_2,R_3) = R_1 \bowtie R_2 - R_3$ on the hypercube illustrated in Fig. 1. The size of each table is $\|R_1\| = 15$MB, $\|R_2\| = 12$MB, and $\|R_3\| = 25$MB. If the estimate [9, 14] of the result size $\|R_1 \bowtie R_2\|$ is 8MB, peer-to-peer collaboration derives an execution plan as (1) send R_2 to R_1, compute $R_1 \bowtie R_2$ at R_1's site, and send R_3 to R_1's site to compute the final result, (2) do (1) but send the result of $R_1 \bowtie R_2$ to R_3 instead, (3) do (1) but start by sending R_1 to R_2, and (4) do (3) but send the result of $R_1 \bowtie R_2$ to R_3 instead. The transmission cost is respectively (1) $\|R_2\| \times 2 + \|R_3\| \times 3 = 99$MB, (2) $\|R_2\| \times 2 + \|R_1 \bowtie R_2\| \times$

$3 = 48$MB, (3) $\|R_1\| \times 2 + \|R_3\| \times 1 = 55$MB, (4) $\|R_1\| \times 2 + \|R_1 \bowtie R_2\| \times 1 = 23$MB. Therefore, peer-to-peer method will select plan (4).

On the other hand, the delegation to centrality method finds a server node eligible to gather data and process the query. Suppose there are three servers with the requisite processing capabilities: for R_1 to be a center, the total amount of data transfer is $\|R_2\| \times 2 + \|R_3\| \times 3 = 99$MB. for R_2, $\|R_1\| \times 2 + \|R_3\| \times 1 = 55$MB, for R_3, $\|R_1\| \times 3 + \|R_2\| \times 1 = 57$MB, and therefore R_2 should act as a center.

A plan computation for the peer-to-peer collaboration is particularly demanding. For example, if 3 nodes R_1, R_2, and R_3 are participating in a join $R_1 \bowtie R_2 \bowtie R_3$, it is necessary to examine 3!/2 sequences of the form due to the commutative property of join operation to determine which one gives the minimum cost. The evaluation of *Example 2* by adjusting to $((R_1 - R_3) \bowtie (R_2 - R_3)) - R_3$ for possible optimization is costly and clearly a brute force approach to solving this problem is of exponential complexity.

An alternative is to designate a node that is relatively central to those providing data for the query, and sending all the data to that node to complete the relational operation [17]. The first step in this delegation procedure is finding the central node. The simplest way to do this is by means of the Floyd-Warshall algorithm [13, 12], a dynamic programming approach for finding the shortest paths between every pair of vertices in the subgraph formed by the nodes involved in the query operation. From the matrix of shortest distances produced by this algorithm, a central node can be chosen by comparing the sums of data times distance for each of the nodes in the subgraph. Floyd-Warshall executes in $O(m^3)$ steps where m is the number of nodes serving as data servers.

3 Delegation to Centrality: Simulation Study

To investigate the effect that a central node exerts on data transmission and network congestion, we have implemented a simulation utilizing the Python NetworkX [26] package. A patent [5] filed by IBM-UK for deriving a hypercube center in linear time does not work for our case because the derived center may not be able to act as a server. The delegation of work must occur at a node equipped with sufficient processing power as well as software and hardware that enables it to process queries. We have developed a new method to find an appropriate center, choosing one of the m-servers from the n-node network.

3.1 Experimental Method

As illustrated in Fig. 2, the simulator generates an n-node random geometric graph [27] with a single connected component in which m-nodes representing data servers are colored red, and the network center (node 39) derived by the Warshall-Floyd method for the entire network is colored yellow. It then constructs a subgraph that connects m-nodes and finds a center (node 47) colored here sky blue. Note that

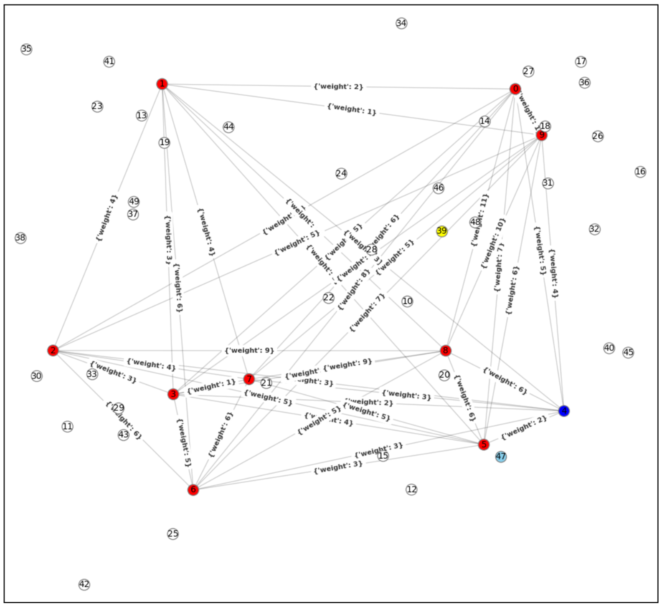


Fig. 2 Random geometric graph and derived centers

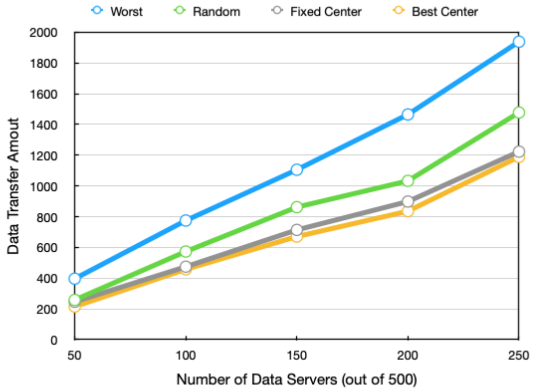
the figure shows a small-scale network with $m = 10$ and $n = 50$ for purposes of illustration. While this is a center of the m-node subgraph, it may not be a data server as the figure indicates. Hence, the simulator reconstructs an m-node subgraph connecting only servers by finding pair-wise shortest distances of m-nodes and applying the Warshall-Floyd method again to find a center. The figure shows the server center colored dark blue. We conducted a series of experiments by setting out different combinations of n and m and different network structures to derive the average total size (product of data size and network hop count) to observe the growth of data circulating in the network. Specifically, for each combination of n and m, we generated 10 different graphs and for each such graph we generated 10 sets of different data sending schemes to average the total size of data transmission. In other words, the measure was derived from 100 simulation experiments.

3.2 Experimental Result

Here we present one result from the experiment set, with the following configuration: n = 500, m = 50, 100, 150, 200, and 250. Data servers are randomly selected from 1/4 of m nodes, and each will have a data segment whose size (in arbitrary units) is drawn from a uniform distribution [10, 100]. The following explains the plot and values in Fig. 3.

(1) The worst (and largest) data amount to be transferred due to the choice of the most distant node for data processing.

(2) The data amount to be transferred by choosing a node at random among m-nodes for data processing.
(3) The data amount to be transferred by selecting the center without the use of data weights.
(4) The best (and smallest) data amount to be transferred by deriving the center that minimizes data transfer by the use of data weights.



m	**(1) worst**	**(2) random**	**(3) fixed**	**(4) best**
50	396.04 (184,4%)	257.82 (120.1%)	245.94 (114.5%)	214.76 (100%)
100	774.60 (169.5%)	572.90 (125.3%)	474.08 (103.7%)	457.12 (100%)
150	1,105.10 (164.9%)	861.78 (128.6%)	713.56 (106.5%)	670.30 (100%)
200	1,464.72 (175.3%)	1,032.70 (123.6%)	878.22 (107.5%)	835.52 (100%)
250	1,937.28 (163.0%)	1,477.52 (124.3%)	1,223.00 (102.9%)	1,189.08 (100%)

Fig. 3 Experiment result with $n = 500$ and $m = 50 \sim 200$

The percentage values in the table are relative to the results in the best cases. The experimental results shown in Fig. 3 highlights a finding that a network center derived from an m-node subgraph, not taking account of the amount of data to be transferred, gives near-optimal performance in every case. The experiments with other variations produced similar results. This outcome could be due to the relatively high capacity of the transmission links, which apparently did not allow for distinguishing transmission times. However, the result we observed could generalize to a network with a large number of participating servers in which the volume of data to be exchanged is reasonably uniform. In this case, the delegation to centrality method could choose a center from m-nodes based solely on distance considerations, not making use of information about the amount of data to exchange. Our finding indicates that in principle the investment made in the server located in the fixed center of m-nodes should be most advantageous if the network structures of the group of data hosting sites are fairly stable. On the other hand, an unsupervised choice of center, i.e., random selection, will perform less well requiring about 20% more data being transmitted.

4 Implementation by Software Design Networking

The objectives of this research are to investigate the performance characteristics of the above method and to develop and deliver the tools for achieving optimal or semi-optimal performance of the set-oriented relational algebra operations in data distributed environments. The tools will be designed to work in random and engineered networks as well as in more constrained environments such as low-bandwidth dynamic networks established across a set of co-operating organizations. Further research is needed to determine the precise conditions of data distribution that favor use of such set-oriented operations either prior to or post query operations. The distributed query optimization in this comprehensive level of work has not been addressed in the existing literature. Therefore, we believe a successful outcome of our research will contribute to the solution of an outstanding problem.

4.1 Systems Framework

Our initial approach is to implement an operating systems environment consisting of a set of independent processes serving virtual data hosts. Communication among these processes can be achieved efficiently by means of a hypercube overlay network implemented with commonly available software-defined networking tools. Distributed applications can be built based on multiple open-source, heterogeneous, relational database systems such as MariaDB, PostgreSQL, Firebird, SQLite3. These databases connect through a network maintained by the OpenFlow [25] environment and the software-oriented Address Resolution Protocol implemented with Ryu [29] component packages. Portability of the system is ensured through the use of the Ubuntu 18.04 operating system built with a Mininet [24] virtual machine.

Our experimental system will create a static hypercube to accommodate a predetermined number of nodes, and will activate an overlay structure by accessing a designated IP address that hosts a specific database system. We have built several

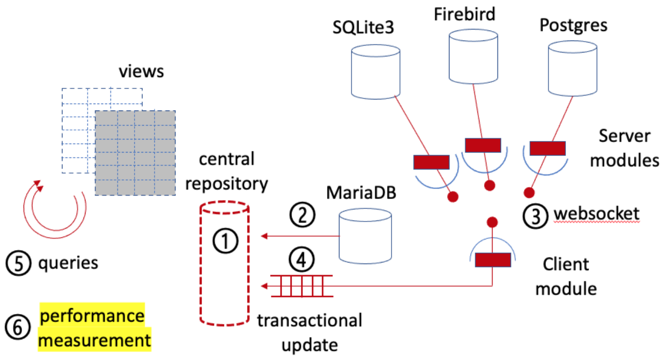


Fig. 4 Distributed data processing application environment

distributed query processing applications designed to gather data from a publicly available data repository system, namely New York City OpenData [11]. Collections of interest in this system include records of parking violations, housing litigation, arrests, complaints, etc. Fig. 4 shows the software structure of our distributed applications: the system will (1) erase central server's common data repository at the start of query execution, (2) deposit central server's data segment to the common repository, (3) communicate with other databases through websocket to access their data segments and obtain them as Json streams, (4) unpack Json streams and insert data segments to the common repository each as a database transaction, (5) execute query as view selections, and (6) report response time spent for steps (1) through (5).

4.2 Preliminary Performance

A preliminary performance measure has been obtained from one distributed application that manipulates 'arrest records' and 'complaint records' stored in OpenData. The two data sets are divided into four databases as shown in Table 1. The queries retrieve the following: (1) the number of arrests from the three boroughs with the most complaints (2) the top 10 offenses complained about in boroughs that do not have the fewest arrests (3) the percentage of arrests per complaint across all boroughs for all racial groups.

The four database are placed in a 16-node (2^4) hypercube and two sets of experiments were done to measure the data preparation time (steps (1) through (4) in Fig. 4) and query response time (step (5) in Fig. 4). The response times in Table 1 indicate the time spent by each of the four central database to complete the above three queries. The experiment showed that the insertion time to Firebird was long compared to other three databases, and that in this kind of light-weight data congestion there would be no significant difference in the database locations in the hypercube. Comparison of performance between a hypercube overlay network and a set of randomly built networks is underway.

Table 1 OpenData's record distributions and response time on hypercube

Database	Data Period	Arrest Row Count	Complaint Row Count	Min. Distance (1 hop away)	Max. Distance (3 hop away)	Query Time
Firebird	Jan. – Mar.	44,822	106,111	226.8 sec	226.9 sec	4.3 sec
SQLite3	Apr. – Jun.	29,959	89,075	43.6 sec	46.2 sec	1.2 sec
MariaDB	Jun. – Sep.	28,593	107,990	41.3 sec	41.2 sec	1.1 sec
PostgreSQL	Oct. – Dec.	37,039	101,253	36.7 sec	38.3 sec	1.3 sec

5 Conclusions

We have discussed a novel method for consolidating data in an engineered hypercube network for the purpose of optimizing distributed query processing. The principle approach works at an application layer to minimize data transfer, measured as the product of data quantity and network distance, by delegating the query processing to a node that is relatively central to the network. The hypercube facilitates computation of the network distance between nodes and the location of a node to serve as the center for any data consolidation operations. We sketched the demonstration effort and described a prototype implementation of a hypercube using Software-Defined Networking to support query optimization in distributed, heterogeneous database applications. At the time of writing, we are engaged in measuring performance and plan to report more results from our ongoing work. We will also incorporate a two-phase exchange method, adapted from well-known semi-join operations [6] to ensure that data transfer between the central node and data server nodes will substantially reduce network congestion [17].

Current research on 5G networks shows the need for computational resources that are neither readily available nor adequate with existing facilities in enterprise cloud environments. IoT entails a shift to edge computing, making use of urban datasets coupled with expertise across disciplines to address challenges in urban environments. We believe that the approach introduced in this paper offers a viable solution to reduce the congestion due to the high volume of data traffic in emerging network applications. Additionally, we believe that our approach will prove useful for collaborative projects that address problems of national importance including health management, urban informatics, and climate science.

Acknowledgement: Authors of this paper are grateful to students in the Senior Design courses offered during the academic year 2020–2021 at the City College of New York. Each of the six teams produced a distributed database application which runs on the hypercube of 2^4-nodes and successfully completed performance experiments to exhibit the result in this paper.

References

1. K. Arabi. Mobile computing opportunities, challenges and technology drivers. In *IEEE DAC 2014 Keynote*, 2014.
2. K. Arabi. Trends, opportunities and challenges driving architecture and design of next generation mobile computing and iot devices. In *MIT MTL Seminar*, 2015.
3. G. Bent, P. Dantressangle, P. Stone, D. Vyvyan, and A. Mowshowitz. Experimental evaluation of the performance and scalability of a dynamic distributed federated database. In *Proceedings of the Second Annual Conference of ITA*, September 2009.
4. G. Bent, P. Dantressangle, D. Vyvyan, A. Mowshowitz, and V. Mitsou. A dynamic distributed federated database. In *Proc. 2nd Ann. Conf. International Technology Alliance*, 2008.
5. G. A. Bent, P. Dantressangle, and P. D. Stone. Optimising data transmission in a hypercube network. Technical report, IBM-UK, 2019.
6. P. A. Bernstein and D.-M. W. Chiu. Using semi-joins to solve relational queries. *J. ACM*, 28(1):25–40, Jan. 1981.
7. P. A. Bernstein, N. Goodman, E. Wong, C. L. Reeve, and J. B. Rothnie, Jr. Query processing in a system for distributed databases (sdd-1). *ACM Trans. Database Syst.*, 6(4):602–625, Dec. 1981.
8. L. Bouganim, F. Fabret, C. Mohan, and P. Valduriez. Dynamic query scheduling in data integration systems. In *16th International Conference on Data Engineering*, pages 425–434, 02 2000.
9. S. Chaudhuri. An overview of query optimization in relational systems. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 34–43. ACM, 1998.

10. H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl. Understanding smart cities: An integrative framework. In *2012 45th Hawaii International Conference on System Sciences*, pages 2289–2297, 2012.
11. City of New York, https://opendata.cityofnewyork.us. *NYC OpenData*.
12. T. Cormen. *Introduction to algorithms*, chapter 15. The MIT press, 2 edition, 2001.
13. R. W. Floyd. Algorithm 97: Shortest path. *Comm. of ACM*, 5(6):345, 1962.
14. H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.
15. Y. Jiang, D. Taniar, and C. Leung. High performance distributed parallel query processing. *Comput. Syst. Sci. Eng.*, 16:277–289, 09 2001.
16. A. Kawaguchi, A. Mowshowitz, A. Nagel, A. Toce, G. Bent, P. Stone, and P. Dantressangle. A model of query performance in dynamic distributed federated databases taking account of network topology. In *Annual Conference of International Technology Alliance in Network and Information Science (ACITA2012)*, September 2012.
17. A. Kawaguchi, A. Mowshowitz, and M. Shibata. Semi-operational data reductions for query processing in highly distributed data environments (extended abstract). In *US-Japan Workshop on Programmable Networking, Kyoto Japan*, November 2020.
18. D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.
19. A. Mowshowitz and G. Bent. Formal properties of distributed database networks. In *Annual Conference of the International Technology Alliance, University of Maryland*, 2007.
20. A. Mowshowitz, G. Bent, and P. Dantressangle. Query optimization in a distributed hypercube database. In *Proceedings of the Third Annual Conference of ITA*, September 2010.
21. A. Mowshowitz, A. Kawaguchi, A. Toce, A. Nagel, G. Bent, P. Stone, and P. Dantressangle. Query optimization in a distributed hypercube database. In *Proceedings of the Fourth Annual Conference of ITA*, September 2010.
22. A. Mowshowitz, A. Kawaguchi, A. Toce, A. Nagel, G. Bent, P. Stone, and P. Dantressangle. Query optimization in a distributed hypercube database. In *Proceedings of the Fourth Annual Conference of ITA*, 2010.
23. A. Mowshowitz, A. Kawaguchi, and M. Tsuru. Topology as a factor in overlay networks designed to support dynamic systems modeling. In *13th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2021), in press*, 2021.
24. Open Networking Foundation, https://mininet.org. *Mininet*.
25. Open Networking Foundation, https://opennetworking.org. *OpenFlow*.
26. N. Organization. *Networkx – Network Analysis in Python*. https://networkx.org.
27. M. Penrose. *Random Geometric graphs*. Oxford University Press, 2003.
28. J. Rothnie Jr, P. Bernstein, S. Fox, N. Goodman, M. Hammer, T. Landers, C. Reeve, D. Shipman, and E. Wong. Introduction to a system for distributed databases (sdd-1). *ACM Transactions on Database Systems (TODS)*, 5(1):1–17, 1980.
29. Ryu SDN Framework Community, https://ryu-sdn.org. *Ryu*.
30. T. Saadawi, A. Kawaguchi, M. J. Lee, and A. Mowshowitz. Secure resilient edge cloud designed network (invited). *IEICE Trans. on Communications*, E103-B(4), April 2020.
31. D. Taniar, C. H. C. Leung, J. W. Rahayu, and S. Goel. *High Performance Parallel Database Processing and Grid Databases*. John Wiley & Son, 2008.
32. A. Toce, A. Mowshowitz, A. Kawaguchi, P. Stone, P. Dantressangle, and G. Bent. Hyperd: Analysis and performance evaluation of a distributed hypercube database databases. In *Proceedings of the Sixth Annual Conference of ITA*, 2012.
33. A. Toce, A. Mowshowitz, A. Kawaguchi, P. Stone, P. Dantressangle, and G. Bent. An efficient hypercube labeling schema for dynamic peer-to-peer networks. *Journal of Parallel and Distributed Computing*, 102:186 – 198, 2017.
34. S. Yi, C. Li, and Q. Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, 2015.
35. I. Yildirim. Query operations in highly distributed environment. Master's thesis, City College of New York, 2014.