QoS-based Data Access and Placement for Federated Systems

Wen-Syan Li Vishal S. Batra* Vijayshankar Raman Wei Han Inderpal Narang

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120 USA
Email: wen@almaden.ibm.com

1 Motivations

A wide variety of applications require access to multiple heterogeneous, distributed data sources. By transparently integrating such diverse data sources, underlying differences in DBMSs, languages, and data models can be hidden and users can use a single data model and a single high-level query language to access the unified data through a global schema.

To address the needs of such federated information systems, IBM has developed the DB2 Information Integrator (II) [1] to provide relational access to both relational DBMSs and non-relational sources, such as file systems and web services. These data sources are registered at II as nicknames and thereafter can be accessed via wrappers. Statistics about the remote databases are collected and maintained at II for later use by the optimizer for costing query plans.

DB2 Information Integrator deploys cost-based query optimization to select a low cost global query plan to execute. Thus, cost functions used by II heavily influence what remote servers (i.e. *equivalent* data sources) to access and how federated queries are processed. Cost estimation is usually based on database statistics, query statements, and the local and remote system configuration, such as the CPU power and I/O device characteristics. DB2 allows the system administrator to specify expected network latency between II and the remote servers. However, existing cost functions do not consider (1) the load on the remote servers, (2) dynamic nature of network latency between remote servers and II, and (3) the availability of the remote sources. As a result, federated information systems cannot dynamically adapt to runtime environment changes, such as network congestions or load spikes at the remote sources. Also, since the query plans are generated via cost-based decision making process, currently, there are no mechanisms to avoid fast but unreliable sources. Furthermore, II optimizes user queries individually rather than optimizing a workload as a whole nor does it consider QoS goals. In some scenarios, it is required to distribute queries among servers for balancing load and differentiating QoS requirement for response time.

2 System Architecture

In [2], we introduce a novel *query cost calibrator (QCC)* to calibrate the cost function based on system availability, process and network latency at the remote sources, and the load at the II nodes. QCC transparently adapts the cost functions to the runtime environment, and indirectly *influences* the federated query optimizer in query planning. QCC enables the selection of a set of appropriate remote sources or replicas with consideration of the runtime environment to improve the *predictability* of query response time. QCC also identifies alternate query plans and recommends load balancing strategies to improve overall response time for the workload and availability.

We have extended the technology developed in [2] by incorporating the notion of QoS into query processing as well as autonomic data placement functionality. We now describe the new system architecture of our prototype (shown in Figure 1). A DB2 II-based federated system is enhanced transparently with three complementary components: (1) a meta-wrapper (MW), (2) a query cost calibrator (QCC), and (3) a data placement advisor (DPA).

The meta-wrapper serves as a middleware between II and wrappers. At the compile time, MW receives queries from II and records (a) the incoming federated queries, (b) the outgoing query fragments, (c) the estimated cost of the federated queries and query fragments, and (d) their mappings to the remote servers. The overhead of logging in the meta-wrapper is inexpensive compared with the federated query processing cost. The log is periodically collected by QCC for analysis.

During the run time, MW records (e) the response time of each query fragment. This information, (a)-(e), is forwarded to QCC for further processing and analysis. Based on the estimated cost at the compile time and actual exe-

* This work was performed when the author was on an assignment at IBM Almaden Research Center. He is currently affiliated with IBM Indida Research Center and can be reached at bvishal@in.ibm.com.

**Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005**

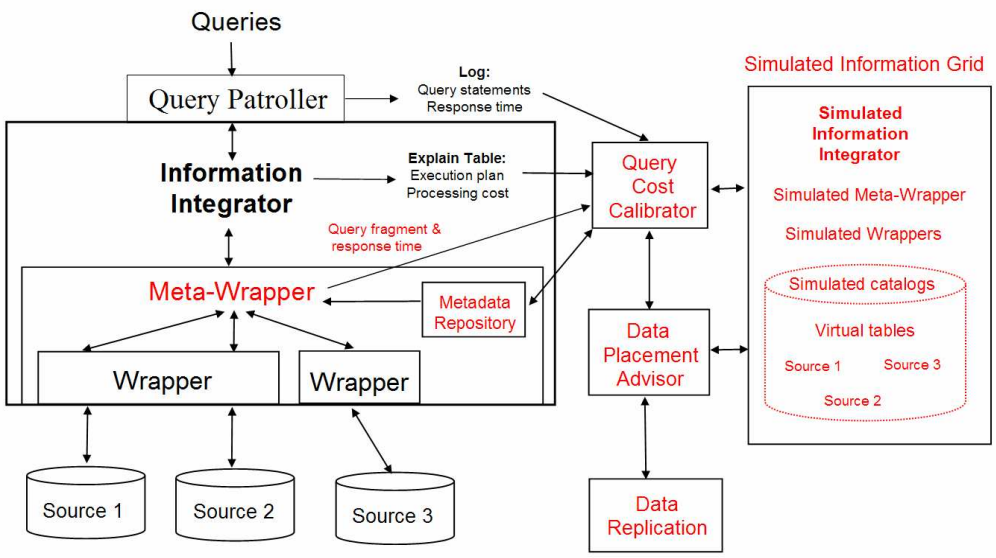

Figure 1: Federated Systems with MW, QCC, and DPA

cution time monitored at the runtime, QCC derives an up-to-date query fragment processing cost calibration factor. Using this factor, QCC can dynamically calibrate the future query estimation cost so that various system characteristics, such as remote system loads and network latency, are implicitly taken into consideration in global query costing. In addition to such transparent statistics collection, QCC uses daemon programs that periodically access remote sources, through MW, to ensure their availability. The daemon programs are also used to derive initial query cost calibration factors by exploring the network latency and processing latency at remote sources.

When wrappers are not able to provide cost estimation or it is not feasible to access remote servers to get necessary database statistics for estimating query processing cost, QCC features a simulated federated system that has the same II, meta-wrapper, and wrappers as same as the original run time system as well as the simulated catalog and virtual tables, to capture database statistics and server characteristics without storing the actual data. The simulated federated system allows QCC to derive alternative query plans and perform "what-if" analysis for query routing and data placement.

3 QoS-based Data Access and Placement

After query compilation at II, only the global query plan with the lowest cost is stored in the `explain` table. When queries are unique, this approach of choosing low cost plans is suitable. However, if there is a large number of similar queries that use the same plan, then the remote servers involved in this plan can get overloaded, rendering the original statistics invalid. To prevent such hot-spots and achieve proper load balance, through the calibration and query routing of QCC, II is enabled to use alternative (maybe not the lowest cost, but close) global query plans in addition to the lowest-cost query plan. QCC also enables QoS aware query routing by selecting query plans that *best match* with specified QoS goals for each user group and service class on alternative servers.

To carry out load balance at the global query level, QCC needs to derive all possible global execution plans (as well as eliminate some less efficient plans). QCC utilizes the simulated federated system to generate all alternative global execution plans and estimate their *calibrated* costs. QCC achieves this by iterating through possible query fragment pairs one at a time at the wrapper level. Calibrated costs of query fragments are used to estimate the cost of the global query plan. The cost of the alternative global query plans are then calibrated by the information integration cost calibration factor as described earlier. Once the calibration costs of all alternative query plans are derived, QCC can eliminate plans that are not promising. Next, QCC identifies plans which have similar costs (i.e. within 20%) and these are executed on different sets of servers. Therefore, QCC identifies groups of plans to recommend to II in a round robin fashion for the given query. By selecting the plans in this way, II distributes the load to multiple servers in a balanced way. Note that, since QCC maintains the server cost calibration factors for all remote sources, it can exclude those remote sources with very high server cost calibration factors from being considered as candidates for query routing destinations.

Additional consideration is needed when there are different QoS requirements for various service classes. In this scenario, selecting a single low cost global query plan and applying this plan to all service classes is not necessarily ideal. In Figure 2, we show an example where all global query processing plans and their calibrated costs are derived for $Q1$ and five alternative plans are eliminated. On the right of Figure 2, we show that there are seven service classes with various response time requirements: 7, 8, 10, 15, 20, 30, and 40. With the current query processing scheme, II will select the lowest calibrated cost global

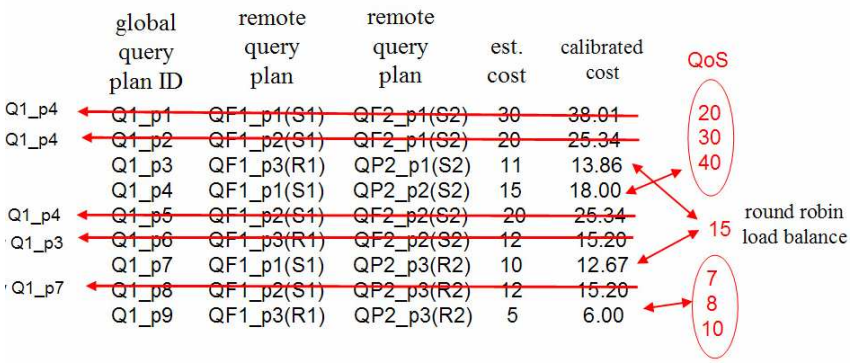

Figure 2: Query Routing with Consideration of QoS Requirement and Load Balance

query processing plan, $Q1\_p9$, for all user groups. As a result, all requests will be routed to servers $R1$ and $R2$. This may result in an overload at $R1$ and $R2$; consequently, users may observer a response much slower than the expected response time, 6. On the other hand, we can see that for those user groups whose data-tier response time requirements for $Q1$ are greater than 20, there is no need to route the queries to $R1$ and $R2$ based on $Q1\_p9$. Their requests can be safely routed to $S1$ and $S2$ (based on $Q1\_p4$) as the expected calibrated response time of 18 will meet their QoS goals. Therefore, instead of assigning the same plan with the lowest cost to all user queries, QCC intelligently assigns plans that best matches the QoS requirement of the service classes, while using the available resources effectively. This assignment scheme allows critical resources to be preserved for the requests that need them most, while satisfying more users' QoS requirements. Note that, in this example, for the service class with the QoS requirement of 15, we have two alternatives: $Q1\_p3$ and $Q1\_p6$. Since the costs of these two plans are close and they are executed in different sets of serves, $Q1\_p3$ and $Q1\_p6$ can be grouped together and chosen in a round robin fashion for this service class as described earlier.

Assignment of global plans based on QoS, since they effect server loads, may change the cost of the global plan. For example, after QCC routes the queries issued by the user groups with QoS requirements of 7, 8, and 10 to $R1$ and $R2$, the actual response time of $Q1\_p9$ may increase from 6 to 8. Thus, QCC re-assigns the query plan $Q1\_p7$ to the requests from the user group whose QoS requirement is 10 upon cost re-calibration.

We have developed a set of guidelines for query routing with considerations to the QoS requirements. The guidelines are summarized as follows:

- For global query plans whose query fragments are executed on the same set of servers, pick the cheapest plan.
- For each federated query with a QoS requirement, pick the global query plan whose calibrated runtime cost best matches the QoS goal.
- Reassign query plans if a cost re-calibration results in failing to satisfy QoS requirements.

Note that the workload of the query (i.e. calibrated cost times the frequency of queries issued in a period) must be greater than a preset threshold value in order for the query to be considered load distribution. If the federated query is inexpensive or seen less frequently compared with other queries, it is not worth deploying the load distribution scheme.

When QCC cannot derive any plan to meet the QoS goals, it alerts the DPA to advise data replication strategies so that the QoS goal can be achieved. DPA derives the server using the simulated information integrator and informs the available replication utility to place and synchronize replicas. In our system, QoS for response time is measured for the whole workload rather than individual queries. Due to the nature of distributed systems, it is not feasible to guarantee the response time of each query. QCC does not trigger data placement advisor immediately if it observe certain QoS goals are not met. Instead, QCC aims at ensuring the average response time for the whole workload meet the QoS goal by ignoring such jitters in system load and network conditions.

In addition to collecting and using cost statistics, QCC also records error messages (if any) from accessing remote servers for assessing their availability and reliability. This information is later used to compute the reliability factor for cost calibration. Consequently, QCC influences II to access not only high performance but also reliable remote servers; adapting to the runtime environment.

We have extended our work in [2] and further enable service class and QoS aware request processing in distributed enterprise information systems, composed of web server, application server, and a data-tier composed of local and remote data servers. Based on the QoS goal for response time specified by each service class and time spent in the web and application server tiers, QCC recommends appropriate query plans and remote servers to meet the end to end QoS goal.

4 Demonstration

We have implemented a prototype system based on the configuration shown in Figure 1. Three servers are made available at IBM facility at IRL (India Research Lab.), SVL (Silicon Valley Lab.), and ARC (Almaden Research Center). A

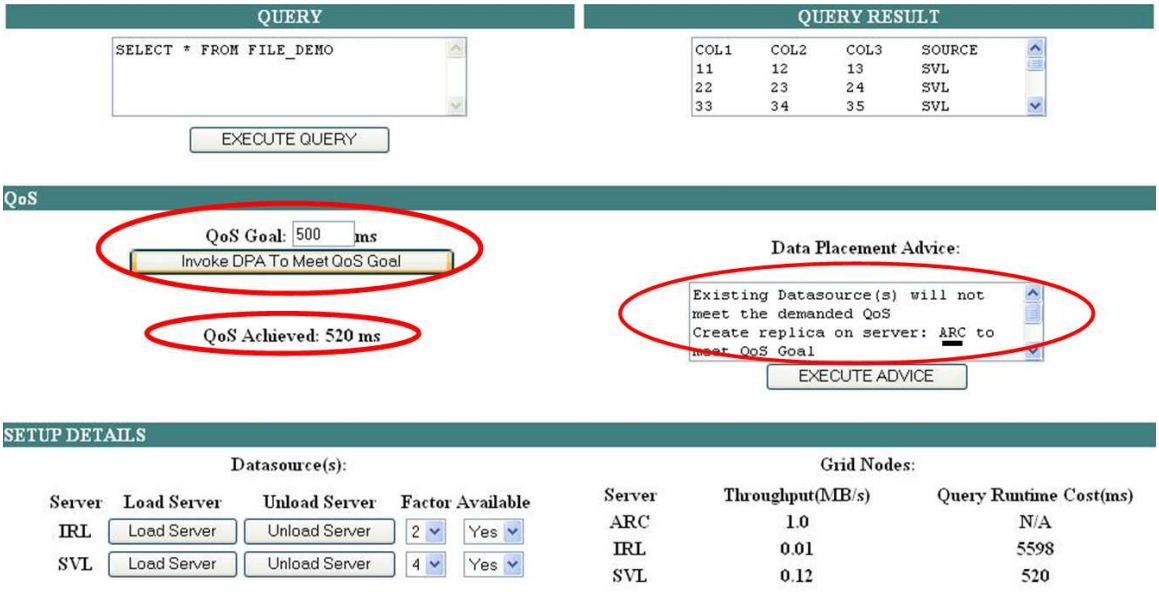

Figure 3: The server at SVL is loaded and cannot meet QoS goal. DPA advises to create a replica at ARC.

DB2 II server is placed at ARC. The meta-wrapper, QCC, and DPA are deployed on the DB2 II server to enable load, network, QoS, and availability aware dynamic federation. In this setting, we are able to load or disable any of these three remote servers.

Figure 3 is a sample screen shot of the demonstration system. On the demonstration console, we can see the federated query statement, query results (and their origin, such as ARC, IRL, and SVL), QoS goals and achieved QoS, data placement advice (if QoS goal cannot be met), and status of three remote servers (including load level, availability, network bandwidth, and estimated response time).

The scenarios of our demonstration are as following sequence:

1. The data is initially available only at IRL. The query is routed to IRL and the response time is measured as more than 5 seconds on average.
2. We now specify a QoS goal as 500ms. QCC detects that the server at IRL cannot meet the QoS goal and DPA is invoked to advise an appropriate server in the grid to create a replica. In this case, DPA advises SVL (i.e. QoS-based data placement).
3. After a replica is created at SVL, the query is routed to SVL instead of IRL in order to meet the QoS goal (as shown in Figure 3).
4. We then add load to the server at SVL so that the QoS goal can no longer be met. Now, DPA advises to create a replica at ARC. After a replica is created at ARC, the query is now routed to ARC. Note that now ARC is the only server that can meet the QoS goal of 500ms response time.
5. Next, we alter the QoS goal by changing it from 500ms to 600ms. QCC now performs a QoS aware routing and directs the query to SVL while reserving ARC for other queries which demand fast response time.
6. Then, we disable the server at SVL. MW detects the server outage at SVL and proactively routes the queries to ARC again.

The system demonstration is built based on commercially available components and runs in real internet environment. With the above scenarios, we are able to demonstrate many novel functions unique on our federated information systems, including (1) query cost calibration, (2) network, load, and availability aware query routing, (3) QoS-based query routing, and (4) QoS-based data placement.

5 Concluding Remarks

In this demonstration description, we highlight the key features of dynamic federation that enable enterprise information systems adapt to runtime environment and QoS goals for response time specified by the users. The current prototype is able to handle simply queries that involve single remote server. We are extending the data placement advisor to handle more complex queries that involve multiple servers with presence of replicas and material views.

References

[1] V. Josifovski, P. Schwarz, L. Haas, and E. Lin. Garlic: A New Flavor of Federated Query Processing for DB2. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002.

[2] Wen-Syan Li, Vishal S. Batra, Vijayshankar Raman, Wei Han, K. Seluk Candan, and Inderpal Narang. Load and Network Aware Query Routing for Information Integration. In *Proceedings of the International Conference on Data Engineering*, 2005.