Analysis of Query Optimization Technique over Web Services

Neha B. Thakare[1], R. R. Shelke[2]

[1]M.E. First Year CSE, HVPM C.O.E.T. Amravati, India,

[2]Prof CSE Department HVPM C.O.E.T. Amravati, India

Abstract: ***The query optimizer is the component of a database management system that attempts to determine the most efficient way to execute a query. The high quality, structured data from Web structured sources is invaluable for many applications. A critical but still largely unresolved question is: how to efficiently acquire the structured information inside Web databases through iteratively issuing meaningful queries? So, a general purpose Web Service Management System (WSMS) that enables querying multiple web services in a transparent and integrated fashion is analyzed. This paper tackles a first basic WSMS problem: query optimization for Select-Project-Join queries spanning multiple web services. Our main result is an algorithm for arranging a query's web service calls into a pipelined execution plan that optimally exploits parallelism among web services to minimize the query's total running time. Surprisingly, the optimal plan can be found in polynomial time even in the presence of arbitrary precedence constraints among web services, in contrast to traditional query optimization where the analogous problem is NP-hard.***

Keywords: Web Service Management System (WSMS), NP-hard, Structured Query Language (SQL), Database Management System (DBMS), Quality of services (QoS), Web Service (WS).

1. Introduction

Web services are rapidly emerging as a popular standard for sharing data and functionality among loosely-coupled, heterogeneous systems. Many enterprises are moving towards service oriented architecture by putting their databases behind web services, thereby providing a well-documented, interoperable method of interacting with their data [1].

Furthermore, data not stored in traditional databases also is being made available via web services. There has been a considerable amount of recent work on the challenges associated with discovering and composing web services to solve a given problem. The Metadata component deals with metadata management, registration of new web services, and mapping their schemas to an integrated view provided to the client. There is a large body of work on data integration, which applies to the Metadata component; we do not focus on these problems in this paper. We are interested in the more basic challenge of providing DBMS-like capabilities when data sources are web services [2]. To this end we propose the development of a Web Service Management System (WSMS): a general-purpose system that enables clients to query multiple web services simultaneously in a transparent and integrated fashion. Overall, we expect a WSMS to consist of three major components; see Figure 1.

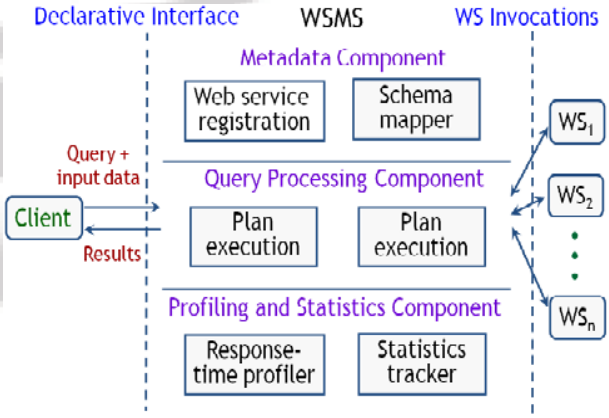

Figure 1: Web Service Management System

Given an integrated view of the schema, a client can query the WSMS through an SQL-like interface [3]. The Query Processing and Optimization component handles optimization and execution of such declarative queries, i.e., it chooses and executes a query plan whose operators invoke the relevant web services. The Profiling and Statistics component profiles web services for their response time characteristics, and maintains relevant statistics over the web service data, to the extent possible. This component is used primarily by the query optimizer for making its optimization decisions [4].

In this paper we take a first step at realizing a complete WSMS: We address the problem of query optimization for Select- Project-Join queries spanning multiple web services. Most web services provide a function-call like interface X → Y where X and Y are sets of attributes: given values for the attributes in X, the web service returns values for the attributes in Y. For example, a web service may take a credit card number and return the card's credit limit.

Due to this very restricted interface, most query processing over web services can be thought of in terms of a "workflow" or pipeline: some input data is fed to the WSMS,

and the WSMS processes this data through a sequence of web services [5]. The output of one web service is returned to the WSMS and then serves as input to the next web service in the pipeline, finally producing the query results. Each web service in the pipeline typically performs operations such as filtering out data items that are not relevant to the query, transforming data items, or appending additional information to each data item. Transformed or augmented data items may be required for further processing of the query (effectively performing a join across web services), or may become a part of the final query result [6].

Deciding the optimal way to perform this pipelining poses several new challenges:

1) Different web services may differ widely in their response time characteristics, as well as in how many output tuples they produce per input tuple on average (henceforth selectivity). Hence different arrangements of the web services in the pipeline may result in significantly different overall processing rates [7]. The optimizer must decide the best arrangement.
2) The web services in the pipeline may not always be freely reordered, i.e., there might exist precedence constraints
3) A linear ordering of the web services in a pipeline may not be optimal. On the other hand, parallelizing all web services without precedence constraints may not be optimal either, since one or more of the web services may vastly reduce the amount of data the others need to process [8].
4) Each web service call usually has some fixed overhead, typically parsing SOAP/XML headers and going through the network stack. Hence some web services support sending data to them in “chunks” rather than one tuple at a time.

Through experiments we found that the response time of a web service often is not linear in the input chunk size, so the optimizer must decide the best chunk size to use. In this paper, we develop new, efficient algorithms that address each of the above challenges to arrive at the optimal pipelined execution plan for a given query over a set of web services [9].

A simple yet significant observation that forms the basis for our algorithms is that the performance of a pipelined plan over web services (the rate of data processing through the plan) is dictated by the slowest web service in the pipeline (referred to as the bottleneck cost metric). In contrast, in a traditional centralized system, the cost of a pipelined plan is dictated by the sum of the costs of the plan operators (referred to as the sum cost metric) rather than by the cost of only the slowest operator [10].

2. Preliminaries

2.1 Query Plans and Execution Model

Query optimization and execution model can be diagrammatically explained as follows:

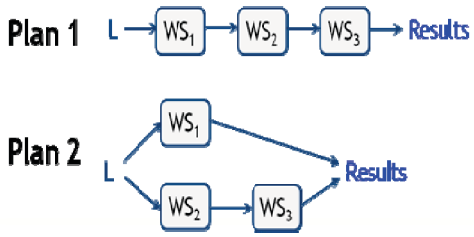

Figure 2 : Query Plans And Execution Model

In general, an execution plan is an arrangement of the web services in the query into a DAG H with parallel dispatch of data denoted by multiple outgoing edges from single web service.

3. Query Optimization

Most query optimizers represent query plans as a tree of "plan nodes". A plan node encapsulates a single operation that is required to execute the query [11]. The leaves of the tree are nodes which produce results by scanning the disk, for example by performing an index scan or a sequential scan. A query is a request for information from a database. Queries results are generated by accessing relevant database data and manipulating it in a way that yields the requested information. Since database structures are complex, in most cases, and especially for not-very-simple queries, the needed data for a query can be collected from a database by accessing it in different ways, through different data-structures, and in different orders.

4. Query Optimization Analysis

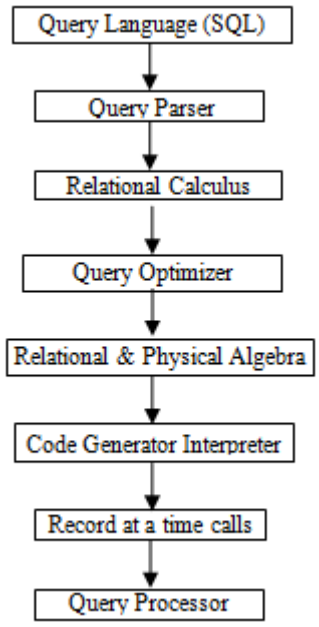

Figure 3: Query flow in DBMS

Each different way typically requires different processing time [12]. Processing times of a same query may have large variance, from a fraction of a second to hours, depending on the way selected. The purpose of query optimization, which is an automated process, is to find the way to process a given query in minimum time [13].

The large possible variance in time justifies performing query optimization, though finding the exact optimal way to execute a query, among all possibilities, is typically very complex, time consuming by itself, may be too costly, and often practically impossible [14]. Thus query optimization typically tries to approximate the optimum by comparing several common-sense alternatives to provide in a reasonable time a "good enough" plan which typically does not deviate much from the best possible result [15].

5. Future Scope

There are several interesting directions for future work:

1) An important next step is to extend our algorithms to allow different input tuples to follow different plans as in leading to even higher overall performance.
2) Our algorithms currently do not incorporate variance or uncertainty in the response times of web services, or more generally, quality of service (QoS) information about web services. It is important to address the problem of finding plans that consistently choose the highest-quality available web services and that adapt to changes in web service response times.
3) Our query optimization algorithm relies on knowledge of web service response times and selectivity's. Hence we need to develop profiling techniques that can accurately track these quantities and detect changes in them. Work on self-tuning histograms may be relevant to track selectivity's.

6. Conclusion

We have analyzed the overall goal of a general-purpose Web Service Management System (WSMS), enabling clients to query a collection of web services in a transparent and integrated fashion. In this paper, we focus on new query optimization issues that arise in a WSMS. Our execution model consists of pipelined query processing over web services, and we derive the cost metric to characterize the cost of a pipelined plan. For this cost metric, we have devised new algorithms to decide the optimal arrangement of web services in a pipelined plan, respecting precedence constraints which will make new revolution in web searching.

References

[1] F. Casati and U. Dayal, Editors. Special Issue on Web Services, Data Eng. Bull., 25(4), 2002.
[2] J.Burge,K.Munagala and U. Srivastava.Ordering pipelined operators with precedence constraints.
[3] M. Ouzzani and A. Bouguettaya. Query processing and optimization on the web. Distributed and Parallel Databases, 15(3):187–218,2004
[4] Chaudhuri,Surajit (1998). "An Overview of Query Optimization in Relational Systems". Proceedings of the ACM Symposium on Principles of Database Systems."
[5] Ioannidis,Yannis(March1996)."Query optimization".
[6] J. Burge, K. Munagala, and U. Srivastava. Ordering pipelined operators with precedence constraints.
[7] F. Casati and U. Dayal, editors. Special Issue on Web Services, IEEE Data Eng. Bull., 25(4), 2002.
[8] S. Chaudhuri and K. Shim. Optimization of queries with user-defined predicates. ACM Trans. on Database Systems, 24(2):177–228, 1999.
[9] Condon, A. Deshpande, L. Hellerstein, and N. Wu. Flow algorithms for two pipelined filter ordering problems. In Proc. of the 2006 ACM Symp. on Principles of Database Systems, 2006.
[10] D. DeWitt et al. The Gamma Database Machine Project. IEEE Trans. on Knowledge and Data Engineering, 2(1):44–62, 1990.
[11] L. Ding and E. Rundensteiner. Evaluating window joins over punctuated streams. In Proceedings of the 2004 ACM Conf. on Information and Knowledge Management, pages 98107, 2004.
[12] D. Florescu, A. Grunhagen, and D. Kossmann. XL: A platform for web services. In Proc. First Biennial Conf. on Innovative Data Systems Research (CIDR), 2003.
[13] D. Florescu, A. Levy, I. Manolescu, and D. Suciu. Query optimization in the presence of limited access patterns. In Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of Data, pages 311–322, 1999.
[14] H. Garcia-Molina et al. The TSIMMIS approach to mediation: Data models and languages. Journal of Intelligent Information System, 8(2):117–132, 1997.
[15] R. Goldman and J. Widom. WSQ/DSQ: A practical approach for combined querying of databases and and the web. In Proc. Of the 2000 ACM SIGMOD Intl. Conf. on Management of Data, pages 285–296, 2000.

Author Profile

Miss Neha B. Thakare has Completed Degree in Bachelor of Computer Science and Engineering from IBSS COET, Amravati, Maharashtra, India. She is pursuing M.E. First Year Computer Science and Engineering in HVPM COET, Amravati, Maharashtra, India.

Prof. R. R. Shelke is working as Assistant Professor, Department of Computer Science & Engineering, HVPM COET, Amravati, Maharashtra, India.