

Optimal Asynchronous Service Scheduling in BISON

Hakan Hacigümüş

James Rhodes

IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA

{hakan,jrhodes}@us.ibm.com

ABSTRACT

In this paper, we present some of our experiences from the architectural design phase of Business Information Analysis service provisioning system, BISON. More specifically we discuss the issues, challenges, and our solution approaches to asynchronously scheduling the service requests in an optimal manner to enhance the user experience in the system.

1. INTRODUCTION

Transforming the data pouring from utterly disparate resources into information to gain insights into business operations is vital for the enterprises to stay competitive. BISON (Business Information Analysis on Demand) is a system that is designed to help enterprises achieve this transformation [1]. In this paper, we present some of our experiences from the architectural design phase of BISON system. We discuss specific challenges and our solution approach to those issues. BISON system is based on Business Insights Workbench (BIW) system that we have developed at the IBM Almaden Research Center. BIW aims at solving the essential problem of transforming the ever increasing amount of data from disparate sources, which are not in a form that could be directly used to support critical business decision making processes, into information that provides insights into the business operations and the competitiveness measures. BIW is a comprehensive data analysis application that allows a knowledge worker to learn from large collections of unstructured documents. BIW was designed in a way that applies domain expertise, through interactions with state-of-the-art unstructured data analysis algorithms and visualization, to provide a global understanding of a document collection. BIW has been used in numerous customer setups, to solve complex problems that require understanding and analysis of very large textual data sets to fulfill the business objectives. Some examples are problem ticket analysis, market trend analysis, patent data analysis, and web information mining. The details of the BIW system can be found in [2] and [3].

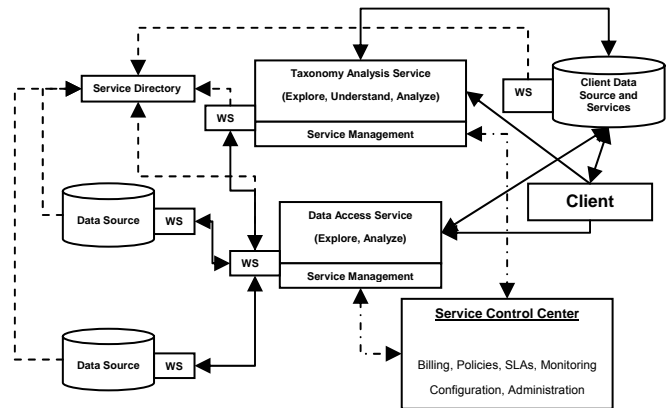


Figure 1. The Architecture of BISON

2. BISON System

We have been developing a business information analysis service provisioning system, BISON, based on BIW system [1]. The short-term goal of BISON is to provide all of the capabilities of BIW system in an SOA environment. The longer-term goal, however, is to develop a repeatable framework and methodology that could be used to create service-oriented systems to process data to gain insights into a given purpose where the data is available from utterly heterogeneous resources. The high-level architecture diagram of BISON system is shown in Figure 1. We use SOA standards as the implementation framework. Communications among the system entities, including the clients, are implemented as Web Services and each web service is defined by using WSDL.

Data Sources may provide structured and unstructured data and metadata information, such as full text indexes. Typical examples for text indexes are indexes created by crawling engines for increased full-text search performance. Data sources expose all the characteristics information, such as name of the source, schema of the database, and types of the data available, and data access

mechanisms as web services. Data sources are registered in the services directory so that they can be discovered by the other system entities. The system architecture allows data sources dynamically join or leave the resource pool, or change the service characteristics.

Data Access Service (DAS) is primarily responsible for querying the data sources to retrieve the data of interest and performing database oriented tasks. It also supports analyze functions over the data sources. The data access service is instrumented to facilitate service provisioning specific tasks.

Taxonomy Analysis Service (TAS) provides all of the enhanced analysis capabilities defined over the taxonomies, that is, explore, understand, and analyze functions. The separation of the data access service and the taxonomy analysis service provides service flexibility and scalability. To be able to perform the functions, TAS relies on the data provided by DAS. Hence, the only way to obtain the data for TAS is through DAS. Once the required data is delivered by DAS, TAS may perform taxonomy analysis specific tasks on the provided data.

Service Control Center oversees the service provisioning tasks in the system. It monitors and aggregates the system management data provided by the individual services.

The client is the consumer of the services. Typically there are two different types of clients in our system. First, the individual users those connect to the service directly by using a web browser. The second category of the clients is the service providers. They use our services to provide additional services to their end-users.

The Client Data Source and Services (CDS) is a temporary data source that is created to maintain the intermediate results between the service calls for the client. It is important to note that, based on the BIW system capabilities, TAS functions may be implemented and executed at the CDS. However, the decision should be made judiciously by considering system specific QoS requirements. This problem motivates the optimization problem we discuss in this work.

3. System Design Issues

In this section we discuss two important design issues that we have encountered in BISON. Although we discuss the issues and the solution approach in the context of BISON, they are applicable to any similar system design in a much larger context.

3.1 Asynchronous Service Calls

In a typical scenario, a knowledge worker uses the functions that are provided by BISON through multiple interactions in an iterative way to gain insights into a document collection. Some of those interactions entail intensive computation; consequently longer running times. We observed that it is possible to start certain interactions

without waiting for the completion of the currently submitted requests for the user. It would dramatically improve the user experience to allow users to move on to different tasks instead of forcing them to wait for an extended period of time. Achieving this requires a means to schedule service requests for different system entities asynchronously.

Asynchronous JavaScript and XML (AJAX) [4] is a web development technique which recently has been increasing in popularity. AJAX promises to create a more rich way to deliver applications via web browsers. The basic premise of AJAX is that clients download a JavaScript engine with a requested web page. As a user interacts with the page, queries are sent to the engine rather than directly to the web server. The engine can then communicate asynchronously with the web server. Additionally, formats such as SOAP or another XML-based web services dialect may be used for communication.

Another benefit of AJAX is that it allows for distribution of tasks between clients and servers. Computational distribution, as well as the asynchronous aspects of the system, is very useful for delivering the services of the BISON platform. Compute intensive jobs such as text clustering can be distributed between both client and server. Documents can be passed asynchronously to the Text Analysis Service via the client as it receives results from the Data Access Service. This allows the client to continue exploring data while waiting for complex tasks to finish.

Unfortunately, a system such as BISON requires many different components to function in optimal conditions. While AJAX provides a framework for distribution of work and asynchronous communication, it does not account for optimal distribution for work events. In order for AJAX to be effective with the BISON system, optimal task partitioning and scheduling among the system services needs to be developed.

3.2 Optimal Service Scheduling

As it was stated above, it is possible to achieve certain tasks by using different set of system entities and the web services exposed by them. A typical example is the TAS. TAS performs taxonomy analysis specific tasks on the data provided by DAS. TAS requests necessary data from DAS and performs the requested tasks on the provided data. Alternatively, the user may request data from DAS and request specific operations to be performed on that data from TAS. To achieve better QoS, however, based on the system's status, some of those operations may be performed by combining the DAS and the CDS instead of directly accessing to TAS. Although this scenario would seem as it is a typical optimal service selection problem, there is a challenging aspect of the problem.

Usually, by the nature of the analysis work, it is not possible to know which services will be requested the system users. Therefore service scheduling decisions can

only be made as the requests from the users are received. This situation makes the optimization process extremely challenging.

We have formulated this problem as an online scheduling problem, which can be solved by following online algorithms approach [5]. The idea behind an online algorithm is that the algorithm does not have access to entire input instance as it makes its decisions. In response to each input portion, the algorithm must generate output, not knowing the future input.

Before we describe our algorithm we provide some definitions and notations. L_c represents the current workload of a particular service entity in the system. Although each service may be delivered by multiple servers in a distributed and/or parallel fashion, this situation is not relevant for our discussion from the notation perspective. P is the performance parameter that characterizes the relevant performance metrics, such as speed, of a given service. Response time $R_s(L_c, P)$ is a function of L_c and P for a particular service s . The response time is defined as the elapsed time between the service request and the completion of the response for the request. Our objective is to optimize the total response time for all of the user requests in an online fashion. The AJAX plays a critical role here. The AJAX technology would allow us to keep the current list of service requests that are being processed for a particular service entity in the system. This information could be used to compute the current workload and the expected response time.

A naive approach to service scheduling would be based on a simple criterion as follows: If $R_{TAS} < R_{DAS} + R_{CDS}$ then assign the request to TAS, otherwise assign it to DAS and CDS combination. The condition essentially checks the expected response times at a particular time and chooses the service or the service combination that has the shorter response time.

We analyze the online approach by using an adversary model that is frequently used to analyze the online algorithms [5]. In the model the adversary sees the outputs generated by the algorithm on previous inputs and generates inputs to force the algorithm to perform poorly. Let us consider the situation where there are many small jobs requests coming in for TAS. Assume that for all of the requests TAS provides the shortest response time. In this case, the algorithm assigns all of the jobs to TAS server. After the sequence of many small jobs, if a very large job request comes in for TAS, the algorithm may not be able to assign the job to TAS because of the current workload and makes a forced decision to choose DAS and CDS combination. However, TAS selection could have been far less expensive in terms of response time had the algorithm offloaded some of the smaller requests from TAS. Similar scenarios could be generated for other combinations as

well. Based on this motivation we propose a heuristic-based online algorithm that enhances the naïve approach.

The heuristic takes a snapshot of the response time R_{TAS}^t before scheduling a task for TAS at time t .

$\delta = R_{TAS}^{t+1} - R_{TAS}^t$ is defined as the difference between two consecutive response time measurements R_{TAS}^{t+1} and R_{TAS}^t .

θ is a pre-defined, system specific threshold value. The algorithm starts scheduling the jobs for DAS and CDS combination when $\theta \leq \delta$ condition is satisfied even TAS has the shorter response time, which is a violation of the original criterion. The intuition behind the heuristic is that it tries to start scheduling the jobs for the alternative services that could achieve the same output before the resource saturation point is reached for the initial service.

A sharper decline, which is captured by δ value, is used as an indicator for the start of that trend and tested by $\theta \leq \delta$ condition. By following this principle, the system keeps the workload in check for TAS thereby conserving resources in anticipation of future service requests. Although we presented the examples based on the system specific services -TAS, DAS, and CDS- the principles can be applied to much more generic service provisioning systems straightforwardly.

4. CONCLUSION

We have presented specific issues that we have encountered during the design phase of a business information analysis service provisioning system, BISON and our solution approach to those issues. We have particularly discussed scheduling the services in an optimal and asynchronous way. The emerging AJAX technology was leveraged for asynchronous service scheduling. We discussed how AJAX could be leveraged to solve the optimal service scheduling problem, which was formulated and studied as an online scheduling problem.

5. REFERENCES

- [1] H. Hacıgümüş, J. Rhodes, S. Spangler, J. Kreulen: BISON: Providing Business Information Analysis as a Service. In Proc. of Intl. Conf. on Extending Database Technology (EDBT) 2006, Munich, Germany
- [2] William F. Cody, Jeffrey T. Kreulen, Vikas Krishna, W. Scott Spangler: The integration of business intelligence and knowledge management. IBM Systems Journal 41(4): 697-713 (2002)
- [3] Scott Spangler, Jeffrey T. Kreulen: Interactive methods for taxonomy editing and validation. CIKM 2002: 665-668
- [4] J. J. Garrett: AJAX: A New Approach to Web Applications, <http://www.adaptivepath.com/publications/essays/archives/000385.php>, 2005
- [5] S. Albers, S. Leonardi: On-line Algorithms. ACM Computing Surveys 31(3es), 1999