On Optimizing Workflows Using Query Processing Techniques

Georgia Kougka and Anastasios Gounaris

Department of Informatics, Aristotle University of Thessaloniki, Greece
{georkoug,gounaria}@csd.auth.gr

Abstract. Workflow management systems stand to significantly benefit from database techniques, although current workflow systems have not exploited well-established data management solutions to their full potential. In this paper, we focus on optimization issues and we discuss how techniques inspired by database query plan compilation can enhance the quality of workflows in terms of response time.

1 Introduction

Workflow management takes the responsibility for executing a series of interconnected tasks in order to fulfill a business goal or implement semi- or fully-automated scientific processes. Typically, most workflow models emphasize the control flow aspect employing well-established models, such as BPEL. Nevertheless, data flow plays a crucial role in the effective and efficient execution and thus is equally significant. Data-centric workflows take a complementary approach and regard data management as a first class citizen, along with the control of activities within the workflow (e.g., [1,4]). An example of strong advocates of the deeper integration and coupling of databases and workflow management systems has appeared in [7]. Earlier examples of developing data-centric techniques of manipulating workflows include the Grid-oriented prototypes in [5,4] and the work in [3]. Those prototypes allow workflow tasks to be expressed on top of virtual data tables in a declarative manner in order to benefit from database technologies; however, they do not proceed to the application of query optimization techniques with a view to speeding up the workflow execution.

The contribution of this work is as follows. We demonstrate how the performance of data-centric workflows can further benefit from techniques inspired by databases. We target workflows that either process unnecessary data or contain services the relevant order of which is flexible, i.e., some activities within the workflow can be invoked in an arbitrary order while producing the same results; we term these services as being commutative. We discuss query optimization techniques that build on top of algebraic laws and the application of those techniques to such workflows with a view to modifying their structure without affecting their semantics in order to improve performance. We focus on fully automated workflows, i.e., workflows that do not require human intervention; in such workflows, the execution of constituent tasks may be cast as (web)

A. Ailamaki and S. Bowers (Eds.): SSDBM 2012, LNCS 7338, pp. 601–606, 2012.

service calls as very commonly encountered in a wide range of workflow management systems. In summary, our proposal aims to bring in a novel dimension in workflow optimization. Currently, the vast majority of workflow optimization efforts deal with scheduling and resource allocation (e.g., [8]); in addition, database-inspired proposals refer to specific applications only (e.g., [2]).

The remainder of the paper is structured as follows. The core part of our proposal is in Section 2. Section 3 contains case studies along with initial insights into the performance gains, and the conclusions appear in Section 4.

2 Database-Inspired Solutions to Workflow Optimization

Workflow optimizations based on structure modifications have not be analyzed to an adequate extent to date. Query optimization techniques are well suited to fill this gap, at least partially. Query plans consist of operators from the relational algebra (e.g., joins, selects, projects), the commutative and associative properties of which are well understood. The theoretic background of query optimization is based on algebraic laws that specify equivalence between expressions. On top of such laws, several optimization techniques can be built. Our idea is to apply similar laws to workflow structures. In other words, we treat workflows as query plans, and the constituent services as query operators in order to allow the application of query optimization rules. In order to apply the techniques suggested hereby, we assume that the precedence constraints between services are known and the input to each invoked activity is a list of data values. Similarly, the service output is another list of values with potentially different number of elements and element size. Such a model is followed by systems such as [6].

Note that in arbitrary business and scientific scenarios, it is common to employ services for data filtering, duplicate removal, transformation and remote method calls, joining inputs, merging inputs, and so on. All those operations correspond to operators such as selects and projects, duplicate elimination, user-defined functions, joins, unions, respectively. More specifically, in our work, we regard the evaluation of a selection predicate, including the case where it contains user-defined functions, as analogous to any workflow service that process a list of data items and produces another list with potentially different number and size of items. Projection complements selection in the sense that selection may filter rows, whereas projection filters columns in database tables. Duplicate elimination in query plans is analogous to services that remove duplicates from lists of items. Grouping is analogous to services that receive a list of input elements, group those elements into groups and process each group as a new element. The join operator is analogous to services that accept multiple inputs and combine them according to some criteria. Furthermore, operators such as unions and intersections directly apply to workflow services operating on datasets.

The fact that workflow services can be mapped to operators implies that common algebraic laws, such as selection and join reordering, commutativity of selections and duplicate elimination, distribution of selections over joins and unions, and pushing selections and projections to operators upstream are applicable to workflows as well. As such, two traditional optimization techniques

that we employ in a workflow context are i) to reorder workflow services so that the services that are more selective are executed as early in the execution plan as possible; and ii) to introduce new filtering services in a workflow, when the data eliminated do not contribute to the result output. Reducing the size of data that we have to process leads to much faster computations. Finally, since the notions of selectivity and cost can be extended for services, we may apply more sophisticated cost-based optimizations, too.

3 Case Studies

We present two representative case studies with real scientific workflows, and we aim to demonstrate the performance benefits when the input data is dirty in the sense that it contains duplicates.

Case Study I: The first case study deals with a simple workflow titled as *"Link protein to OMIM disease"*, which is shown in Fig. 1[1]. Its purpose is to find diseases that are related to certain user-defined keywords. The first two services, *search* and *split OMIM results* correspond to the processes of searching and linking the input to a set of diseases from the OMIM human diseases and genes database[2], and presenting the results as individual lists. When a disease is found, it is extracted and then labeled, with the help of the *extract diseases from OMIM* and *label OMIM disease* services, respectively. The labeling service performs data transformation only, so that XML tags are inserted. The next two services in the workflow are *flatten list* and *remove duplicate strings*. The former removes one level of nesting, whereas the other performs duplicate elimination. If we want to apply a string elimination service to data that appear more than once, the procedure of flattening list is a necessary task because orders datasets. These services are very important because of position and role they have to this workflow.

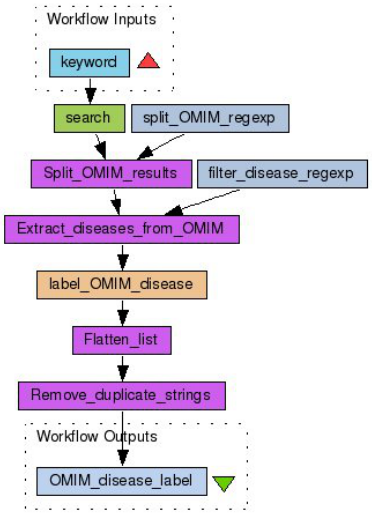

Fig. 1. A workflow that links proteins to diseases

Based on the above description, we can easily deduce (even if not stated explicitly in the workflow description) that the flattening and duplicate elimination services are commutative with the labeling one. In addition, duplicate elimination is the service that may lead to the most significant reductions in the data set manipulated by the workflow, i.e., it is the one with the lower selectivity; actually, the labeling service does not filter data at all and has selectivity 1. As

[1] Taken from http://www.myexperiment.org/workflows/115.html
[2] http://www.ncbi.nlm.nih.gov/sites/entrez?db=omim

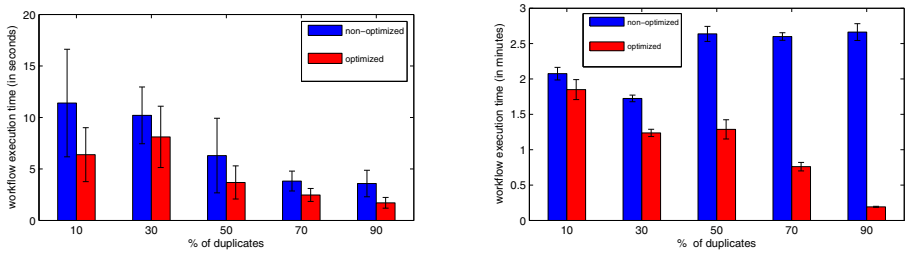

Fig. 2. Performance improvements with 100 Protein/Gene IDs as input for a local workflow (left) and for a workflow accessing a remote service (right)

such, we can improve the execution time if we apply duplicate elimination just before the *extract diseases from OMIM*, in the spirit of optimizations discussed in Sec. 2, or even earlier, i.e., just after the input submission. More specifically, we assume a simplified version of the workflow, where the workflow starts with the *split OMIM results* service and the workflow runs locally only. We consider two flavors, a non-optimized and an optimized one; the non-optimized one performs labeling before duplicate elimination, whereas the optimized one reorders the services and performs duplicate elimination at the very initial stage. Although the modification is simple, there are tangible performance gains. We experimented with an input of 100 OMIM records and a variable proportion of duplicates. Fig. 2 (left) summarizes the results. We used the Taverna 2.3 workbench environment on a Intel Core(TM) 2Duo T7500 machine with 3GB of RAM. The results correspond to the average and the standard deviation of 10 runs after removing the two highest values as outliers to decrease standard deviations. The main observation drawn from the figure is that, if duplicates exist in the input, the decrease in the response time can be higher than 50%.

Case Study II: In the second case study, we experimented with a more intensive subworkflow of a workflow named *"Get Kegg Gene information"*[3]. This sub-workflow gets as input a list of KEGG Genes IDs and according to this list extracts from Kyoto Encyclopedia of Genes and Genomes (KEGG)[4] database information relating to these KEGG genes, such as Gene Description. For example, an input of KEGG Gene ID could be *hsa:400927* which corresponds to the *"hsa:400927 TPTE and PTEN homologous inositol lipid phosphatase pseudogene"* description. The first service of this sub-workflow is *split by regex* and it is necessary in order to reformulate the input data for the workflow services that follow. The key service of this workflow is *get gene description GenomeNet*. This workflow service corresponds to the process of linking KEGG Gene IDs with KEGG Genes database and presenting a brief description for each of the genes. The *get gene description GenomeNet* service is followed by other two services, *merge descriptions* and *remove nulls*, which are responsible for merging the nested content of the previous output and eliminating possible null outputs,

[3] Taken from http://www.myexperiment.org/workflows/611.html
[4] http://www.genome.jp/kegg/

respectively. This service, for each KEGG Gene ID input, connects with the KEGG database in order to search and return the corresponding description of each gene, which is quite time-consuming procedure. In case of having an input data set consisting of several duplicate values, we could assume that the required time to execute a large-scale data set is significantly high. This provides room for optimization, since we can modify the workflow process so that *get gene description GenomeNet* service connects and extracts only unique values of KEGG Gene IDS. Therefore, we can use the techniques used for inserting selections in query plans to perform this optimization task. Specifically, we introduce a new workflow service *remove duplicate gene ids* which has the role to eliminate duplicate values, before the *get gene description GenomeNet* service. In this manner, the service will contact the database only for unique values without repeating the same costly requests. Fig. 3 and 4 depict the original and the optimized version of the sub-workflow examined, respectively.

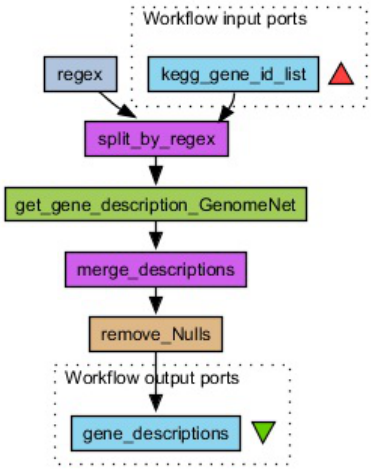

Fig. 3. A sub-workflow of *"Get Kegg Gene information"* that provides description of KEGG Genes

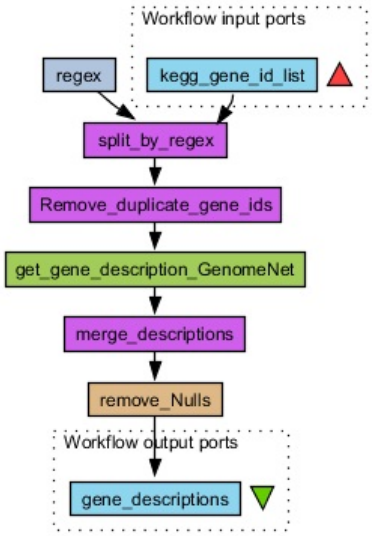

Fig. 4. An optimized sub-workflow that provides description of KEGG Genes

The results shown in Fig. 2 (right) support our intuition that the performance benefits are far more significant and the optimized version runs several times faster. In the second case study, we experimented with a more intensive workflow that involves calls to a remote service, and we inserted duplicate elimination before the service invocation. Specifically, the workflow execution cost is improved gradually while the duplicates of an input data set increases. This means that, for an input data of 100 Gene IDs, which consists of 70% and 90% duplicated values, the optimized version of *"Get Kegg Gene information"* workflow results in 70% and over 90% decrease on response time, respectively.

4 Conclusions

In this paper we discussed the application of query optimization techniques to workflow structure reformations. Our methodology can yield significant performance improvements upon existing data-centric workflows; our preliminary quantitative results in workflows that receive as input data containing duplicates provide promising insights into this aspect.

Our work can be extended in several ways; actually we just scratched the surface of the potential of query optimization techniques for workflows. The most important directions for future work are the development of workflow management systems that fully implement our proposal in line with the points mentioned above, the thorough assessment of performance improvements in a wide range of realistic scenarios, and the investigation of optimization opportunities in more complex workflow patterns. Finally, note that query optimization techniques that seem relevant to workflows are not limited to reordering of commutative services and insertion of early duplicate removal. Complementary aspects include the investigation of equivalent execution plans of different shape and the choice of the physical implementation of logically equivalent workflow services.

References

1. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Modeling, ch. 23, pp. 503–531 (2009)
2. Dayal, U., Castellanos, M., Simitsis, A., Wilkinson, K.: Data integration flows for business intelligence. In: EDBT, pp. 1–11 (2009)
3. Ioannidis, Y.E., Livny, M., Gupta, S., Ponnekanti, N.: Zoo: A desktop experiment management environment. In: Proceedings of 22th International Conference on Very Large Data Bases, VLDB 1996, Mumbai (Bombay), India, September 3-6, pp. 274–285. Morgan Kaufmann (1996)
4. Liu, D., Franklin, M.: The design of griddb: A data-centric overlay for the scientific grid. In: VLDB, pp. 600–611 (2004)
5. Narayanan, S., Catalyrek, U., Kurc, T., Zhang, X., Saltz, J.: Applying database support for large scale data driven science in distributed environments. In: Proc. of the 4th Workshop on Grid Computing (2003)
6. Oinn, T., Greenwood, M., Addis, M., Alpdemir, N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10), 1067–1100 (2006)
7. Shankar, S., Kini, A., DeWitt, D., Naughton, J.: Integrating databases and workflow systems. SIGMOD Rec. 34, 5–11 (2005)
8. Xiao, Z., Chang, H., Yi, Y.: Optimization of Workflow Resources Allocation with Cost Constraint. In: Shen, W., Luo, J., Lin, Z., Barthès, J.-P.A., Hao, Q. (eds.) CSCWD. LNCS, vol. 4402, pp. 647–656. Springer, Heidelberg (2007)