Getting Code Near the Data: A Study of Generating Customized Data Intensive Scientific Workflows with Domain Specific Language

Ashwin Manjunatha[1], Ajith Ranabahu[1], Paul Anderson[2] and Amit Sheth[1]
[1]Ohio Center of Excellence in Knowledge Enabled Computing (Kno.e.sis)
Wright State University, Dayton, Ohio 45435
Email: { ashwin, ajith, amit }@knoesis.org
[2]Air Force Research Laboratory, Biosciences & Protection Division
Wright-Patterson AFB, Dayton, Ohio 45433
Email:paul.anderson2@wpafb.af.mil

Abstract

The amount of data produced in modern biological experiments such as Nuclear Magnetic Resonance (NMR) analysis far exceeds the processing capability of a single machine. The present state-of-the-art is taking the "data to code", the philosophy followed by many of the current service oriented workflow systems. However this is not feasible in some cases such as NMR data analysis, primarily due to the large scale of data.

The objective of this research is to bring "code to data", preferred in the cases when the data is extremely large. We present a DSL based approach to develop customized data intensive scientific workflows capable of running on Hadoop clusters. Our DSL has features to facilitate auto-generation of a Web service front end. These services can be used along with existing service oriented workflow systems. Biologists can use our approach either to implement complete workflows or expose mini workflows as services, all without any knowledge of the underlying complications of the Cloud environment.

1 Introduction

The development of high-throughput biological experimental techniques (e.g., gene chips, mass spectrometry, nuclear magnetic resonance) has lead to the development of the data-driven fields of genomics, proteomics, and metabolomics. The resulting datasets require intensive signal processing and multivariate data analysis for interpretation of experimental results. Typically, large scale NMR data sets are analyzed as follows: (1) standard post-instrumental processing of spectroscopic data; (2) quantification of spectral features; (3) normalization; (4) scaling; and (5) multivariate statistical modeling of data. Each one of these computationally intensive steps can be further subdivided into individual algorithms, where the researcher must select from several competing algorithms. Selecting the best technique for a given task is dependent on the goals of analysis, and therefore, any analysis platform must be robust and flexible. In addition, the parallel nature of processing steps nor the consequences of working with large datasets is seldom realized.

The state-of-the-art for many biologists is to use service oriented scientific workflow approaches. A service-oriented workflow system uses Web services as the atomic building blocks [9]. Successful application of service oriented workflows in NMR-based metabolomics data analysis has been studied by the authors [4].

However our study also revealed several cases where a service oriented approach is either not applicable or sub-optimal. For example some institutions have strict regulations on where the data can be transferred and processed. Even when there are no institutional regulations, transferring large amounts of data is time consuming and in some cases practically impossible. There is a clear need for an alternative approach to create local workflows efficiently and economically.

This work presents a new approach to develop customized data intensive scientific workflows using a Domain Specific Language(DSL) designed for the life sciences domain. The objective of this approach is to *bring the code to the data* rather than *transferring the data where the code is*, the common pattern in service-oriented scientific workflows. We use NMR-based metabolomics data analysis as a use case and present our preliminary study in using a DSL in this space.

The rest of this paper is organized as follows. We present our motivation in Section 2 and discuss the usage patterns in Section 4. The DSL is presented in Section 3 and fol-

lowed by a detailed discussion in Section 5. We present our conclusion in Section 6.

2 Motivation

The primary motivation for this research stems from the numerous complications that arise due to the large amount of data produced by scientific experiments.

- Some institutions (say the Military) have strict regulations on where the data is moved and processed for security and privacy reasons.
- A biologist may want to create a program that runs locally or use the same Cloud (say Amazon) for all the data processing to avoid expensive data transfers. Public Clouds charge for network bandwidth and storage. Given the maximum available bandwidth of todays Internet, it is neither efficient nor cost effective to transfer large amounts of data over the Internet [1]. Alternate ways of data transfer such as shipping hard disks are too cumbersome or time consuming for rapid experimentations.
- Some scientific experiments at their earliest phases, generate extremely large amounts of data. However, the subsequent phases may rely only on a small fraction of this data. It is beneficial to pre-process(e.g. filter) this data near the point of generation to save time and the subsequent data transfer costs.

Apart from these data related concerns, letting a biologist create the relevant programs on their own saves time and improves accuracy by reducing the *number of hops*. Typically the biologists work along with developers who are not domain experts. The process of building the workflow involves several cycles of implementation, testing and verification by the domain experts. Using specialized (biology specific) operators makes creating such a workflow faster by enabling the biologist to directly convert his/her intentions to a workflow. Furthermore this shields the biologists from the complications of the underlying distributed computing environment.

We are motivated to pursue this research by seeing the number of applications that would benefit from this type of technology. For example the Air Force Research Lab (AFRL) conducts many wet lab experiments on biological samples but requires the data to be processed in-house. The biologists have to create either individual services or complete workflows that utilize the in-house computing resources. A disciplined methodology that enables these biologist to create their workflows faster and conveniently is indeed a valuable tool.

3 A DSL for Biologists

We now present the details of our first attempt in creating a DSL suited for biologists. This DSL is created by restricting the Ruby base language. It provides abstractions on top of Apache Pig, a platform for analyzing large data sets over the map-reduce framework, Hadoop [3]. Due to its underlying map-reduce architecture and its fault-tolerant file system, Hadoop is ideal for analyzing large spectroscopic data sets. The layered architecture of the implementation is illustrated in Figure 2.

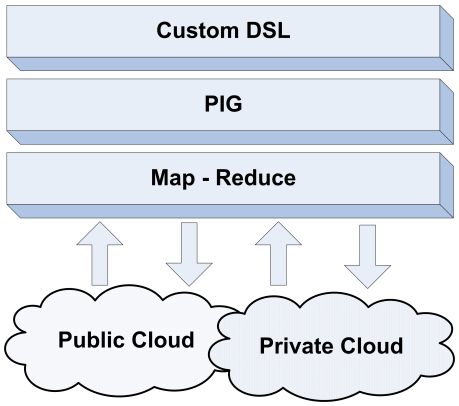


Figure 2: Layered Architecture of the Implementation

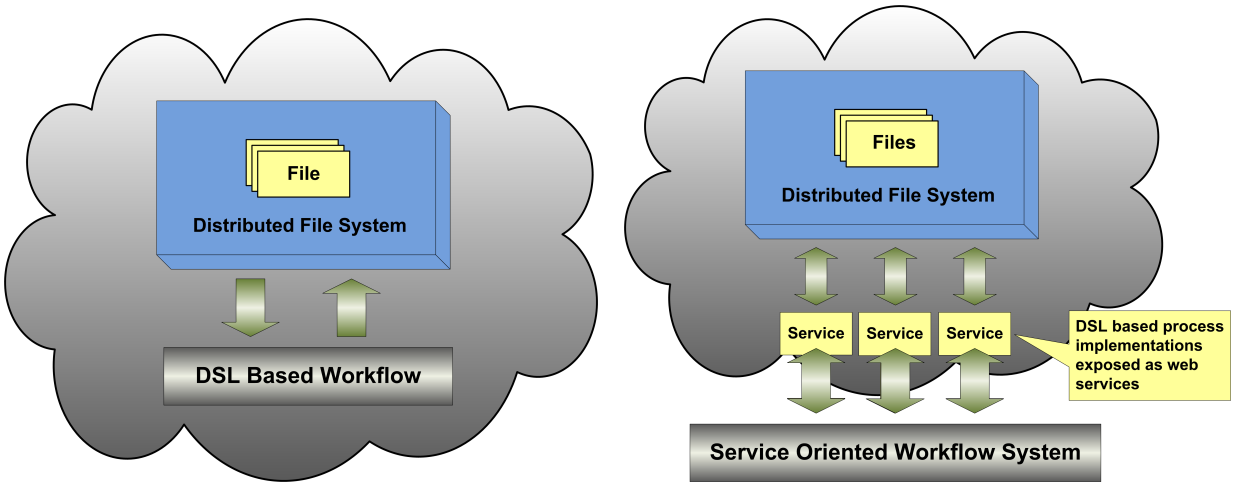
A typical workflow for a NMR data processing consists of one or more of the following steps.

- Loading data (csv, excel, text etc)
- Filtering (range filtering, value based filtering)
- Sorting (ascending , descending with respect to a column)
- Simple statistical functions (max,min, average)
- Signal processing algorithms (sum normalization, auto scaling)
- Writing data (multiple formats)
- Data transformations

Listing 1 outlines a simple *mini workflow* where a data file is loaded, filtered, sum normalized and written back to a new file. The variables raw_data_file and $normalized\_data\_file$ represent the input and the output files respectively. Other function references are self explanatory.

Listing 1: Filtering and Sum normalization implemented using the DSL

```
# load data
original_data = load_data_from_csv(
  raw_data_file)
# filter out a range
filtered = range_filter(
  {:min=> 20, :max => 50},
    original_data)
# sum normalize
normalized = sum_normalize(filtered)
```



(a) Using the DSL to create a complete workflow

(b) Using the DSL to create a individual services that can be used with a service oriented workflow system

Figure 1: Usage patterns for a mini workflow DSL

```
# write the file
store(normalized_data_file,
        normalized)
```

In order to contrast the effort in implementing this in PIGLatin, Listing 2 shows one of the simplest hand written PIGLatin scripts. This script implements sum normalization.

Listing 2: Sum normalization implemented using the PIG

```
A = LOAD '$filename' USING PigStorage(',')
  AS (colnum:int, value:double);
B = GROUP A BY colnum;
C = FOREACH B GENERATE group,
        SUM(A.value);
D = COGROUP A by colnum inner,
    C by $0 inner;
F = FOREACH D GENERATE group,
   FLATTEN(A),FLATTEN(C);
G = FOREACH F GENERATE $0,($2/$4)*100;
STORE G INTO '$filename_processed'
  USING PigStorage(',');
```

There are two observations from these code comparisons.
(1) The PIGLatin script is not intuitive, i.e. its not obvious from the script as to its function.
(2) Creating the PIGLatin script requires a different pattern of thinking and reasoning that needs to be obtained through practice.
It is clearly intuitive for the biologist to follow the first script rather than the second.

4 Usage patterns

There are two possible usage patterns.
(1) Use the DSL to create a complete workflow. This is useful when there are preset operations to be performed over the data and minimum number of branches.
(2) Use the DSL to implement services that can be used in a service-oriented workflow system such as Taverna [5]. The biologists may benefit from being able to expose individual functionality as services and then assembling them via a service-oriented workflow system. Using an existing tool such as Taverna helps in taking advantage of the well established community and third party tool sets. The services may need to be implemented over the same cluster to avoid data transfers.
Figure 1 illustrates these two usage patterns.

5 Discussion

There are advantages and disadvantages of using a DSL. We now outline some of the advantages of this particular approach as well as potential limitations.

5.1 Convenience

Unlike PIGLatin [6] which is not targeted towards a particular domain, this custom DSL is comprehensive to a biologist. Sensible naming of operations and simpler syntax allows biologists to create workflows with ease. Also they are shielded from the intricacies of the underlying Hadoop infrastructure as well as the Web service machinery. This is especially important when experimental data needs to be rapidly processed often with minor variations to the workflow.

5.2 Tooling Requirement

Although a text editor is sufficient for the creation of this script, graphical user interfaces are a welcome addition. A drag and drop style interface similar to Yahoo! pipes [1] is being developed. Most importantly tooling for automatic creation of Web Services and script deployment is necessary for the success of this system. The Web service creation facility is critical in order to enable the second use case discussed in Section 4.

5.3 Provenance and Metadata

A very important addition that can be facilitated via the DSL is automatic addition of provenance and other meta data. Sahoo et al. [7] outline the importance of provenance metadata in the e-science context. The DSL translation can be augmented to automatically incorporate provenance data to the generated data sets, transparent to the users.

5.4 Efficiency

The primary objective of the introduction of this DSL is to bring convenience to biologists rather than creating the most optimal executable code. Since the DSL implementation is based on an abstraction over PIGLatin, the translation of the DSL to executable map-reduce program requires multiple translations. The efficiency of the generated code may deteriorate during these transformations. The convenience of using the DSL far out weighs the impairment to performance in the generated executables.

We follow the argument that it is economical to upgrade the hardware than using human effort to optimize the code [2]. Practitioners generally agree that although there are some cases where code optimization is necessary, it is economical to just upgrade the hardware for many performance issues.

6 Conclusion

Service oriented scientific workflows become less applicable when the data size increases. Apart from the expense in data transfers other constraints such as organizational regulations prevent biologists from using service oriented workflows. DSL based custom workflows are an alternative in this case. Applying a DSL to NMR data processing in metabolomics has shown that this approach is indeed useful and feasible.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[2] J. Atwood. Hardware is Cheap, Programmers are Expensive, 2008. Available online at http://bit.ly/avyNiN - Last accessed Spt 3rd 2010.

[3] A. Bialecki, M. Cafarella, D. Cutting, and O. OMalley. Hadoop: a framework for running applications on large clusters built of commodity hardware. 2005. Avialable online at http://hadoop.apache.org.

[4] A. Manjunatha, A. Ranabahu, P. Anderson, S. S. Sahoo, M. Raymer, and A. Sheth. Cloud Based Scientific Workflow for NMR Data Analysis. 18th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB), 2010. Conference poster.

[5] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, M. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 2004.

[6] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig Latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM, 2008.

[7] S. Sahoo, A. Sheth, and C. Henson. Semantic provenance for eScience: Managing the deluge of scientific data. *IEEE Internet Computing*, pages 46–54, 2008.

[8] R. Stewart. Performance & Programming Comparison of JAQL, Hive, Pig and Java. Technical report, Heriot-Watt University, March 2010. Abstract of Results from MEng Dissertation.

[9] B. Youakim. Service-Oriented Workflow. *Journal of Digital Information Management*, 6(1):119, 2008.

[1] http://pipes.yahoo.com/pipes/