MapReduce Operations with WS-VLAM Workflow Management System

Baranowski, M.; Belloum, A.; Bubak, M.

DOI
10.1016/j.procs.2013.05.449
Publication date
2013
Document Version
Final published version
Published in
Procedia Computer Science

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 2599 – 2602

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

MapReduce operations with WS-VLAM workflow management system

Mikolaj Baranowski[a,*], Adam Belloum[a], Marian Bubak[a,b]

[a]*Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands*
[b]*AGH University of Science and Technology, Department of Computer Science, Mickiewicza 30, 30-059 Krakow, Poland*

Abstract

Workflow management systems are widely used to solve scientific problems as they enable orchestration of remote and local services such as database queries, job submission and running an application. To extend the role that workflow systems play in data-intensive science, we propose a solution that integrates WMS and MapReduce model. In this paper, we discuss possible solution of combining MapReduce and workflow applications, we describe the implementation of chosen solution based on metaprogramming approach in Ruby programming language and evaluate it with an example of word count application.

Keywords: scientific workflow; Ruby; MapReduce; Hadoop; parallel processing

1. Introduction

Workflow Management Systems (WMS) proved to be an easy and efficient way to describe complex systems and to be adaptable to use new technologies. Recently, we can observe an explosion of data and data intensive applications are going to play the more important role in a science. As authors of [1] point out that today's computers has relatively low I/O performance and algorithms should be designed to make computations close to the place where data is stored since any data transfer is very costly [2]. MapReduce is a programming model that has capabilities of processing large volumes of data and meet these new challenges. WMSs that can describe well scientific procedures have to find their place in the data-centric research.

The main objective of our work is to provide access to MapReduce framework from the WMS level. It would enable an access to data stored in a MapReduce-oriented storage like Hadoop Distributed File System (HDFS). The result of this investigations will be implemented using WS-VLAM workflow system as it can be easily extended due to its modular structure [3].

2. Background

The MapReduce [4] programming model was designed for processing large datasets by Google. In Hadoop [5] framework which is an open source implementation of this model, Map/Reduce operations are defined in Java programming language or, by using Hadoop Streaming interface, by any executable. It implies that almost any

*Corresponding author. Tel.: +31-20-525-7562 ; fax: +31-20-525-7490 .
E-mail address: m.p.baranowski@uva.nl.

Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.449

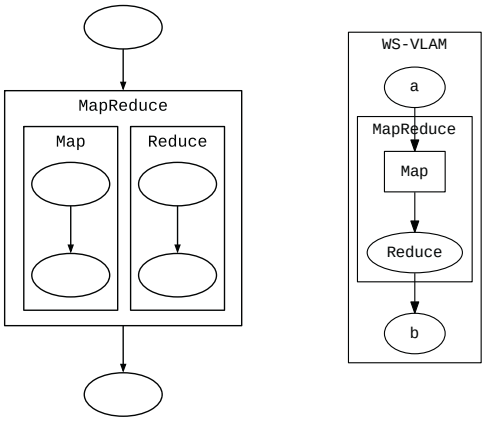

Fig. 1. (a) An example of a workflow that defines Map/Reduce operations. It consists normal workflow tasks – rounded shape, and MapReduce components. Map/Reduce operations are defined as separate subworkflows, on a graph they contain two sequential tasks each. The main MapReduce component groups Map/Reduce subworkflows. (b) An example of WS-VLAM workflow that defines Map/Reduce operations. Nodes `a` and `b` are tasks that send and receive data from the MapReduce component. The `Map` node contains a Ruby script which defines Map operation and the `Reduce` node a name that refers to the applicable implementation.

programming language can be used to define Map/Reduce operations. Pig [6] and Sawzall [7] are tools created by Yahoo and Google to process large number of records with the MapReduce methodology. In Pig framework (built on top of Hadoop), MapReduce jobs are modeled in Pig Lating – a SQL-like declarative language. In Sawzall in turn, there is a Domain Specific Language (DSL) also named Sawzall that allows users to specify Map operations.

In [8], authors are focused on a solution that enriches Kepler with the universal interface to MapReduce. In order to execute MapReduce operations on Hadoop, the MapReduce actor has to be created. It consists two separate sub-workflows that are responsible for two phases - the Map and the Reduce (Fig. 1a). The main advantage of this solution is a fact that Map/Reduce operations are defined in a same way as the rest of the workflow. It makes the model easy to understand for workflow users and allows to use existing WMS modules to define Map/Reduce operations. During execution, Kepler engine and MapReduce subworkflows are distributed to working nodes to perform MapReduce tasks. In the experiment authors measured execution times of the word count application and compared them with the performance of the application that does not use Kepler engine. The native Java implementation turned out to be six times faster than the elaborated solution. Authors conclude that this overhead is due to Kepler engine initialization and workflow parsing. This method of integrating MapReduce with WMS has other disadvantage – some MapReduce implementations – that do not have Hadoop features and wont let to deploy Kepler engine on working nodes – might be unable to integrate. There are also attempts to introduce programming languages to workflow modeling. SMAWL [9] which is is a workflow language based on Calculus of Communicating Systems and application model based on Ruby programming language [10] are notable examples.

When designing our application, we followed notable features from different solutions as the use of DSLs in MapReduce frameworks, metaprogramming techniques in workflow modeling and the specific ideas as the way of choosing Aggregators in Sawzall or an idea of MapReduce workflow tasks from Kepler+Hadoop integration.

3. Enhancing WS-VLAM with MapReduce

WS-VLAM [3] is WMS which covers the entire lifecycle of scientific workflows from design, execution phase to sharing and reuse complete workflows and their components. The aim of our work is to integrate WS-VLAM application with MapReduce framework in such a way that a definition of MapReduce operations will be included in a workflow description and WS-VLAM workflow engine will cooperate with MapReduce environment.

Integrating MapReduce tools as single notes/tasks with WMS would not require that tight integration between MapReduce framework and WMS as Kepler+Hadoop solution. The Map/Reduce operations would be specified

Listing 1. Map operation of word count application

```
require 'mrtoolkit'
class MapWords < MapBase
  def declare
    field :text
    emit :word
    emit :counter
  end
  def process(input, output)
    out = []
    input.text.split.each do |w|
      output = new_output; output.word = w
      output.counter = 1; out << output
    end
    out
  end
end
```

in the most efficient way, however, it would cause inconsistency in the workflow description – Map/Reduce definitions would be created using external tools. In our solution, we follow observations made by the authors of Sawzall [7]. They point out that efficiency is more important in Reduce operations. Map operations spend most of the time waiting for I/O events and they are frequently modified by MapReduce users when Reduce operations, in turn, are rarely changed. It justifies defining Map operations in less efficient way using a metaprogramming approach. We introduce two kinds of workflow tasks – the first one consists a definition of a Map operation in scripting language. The second specifies a Reduce operation by providing the name that refers to the existing implementation of Reduce operation. It has to be clear that these two tasks form a one MapReduce specification.

4. Implementation

Apache Hadoop [5] was chosen as a MapReduce framework. Thanks to Hadoop Streaming interface, we were able to define Map operations using the MapReduce Toolkit from New York Times – MRToolkit [11]. It uses metaprogramming abilities of the Ruby programming language to express Map and Reduce operations. We define Map operations as a single Ruby class that contains a definition of record fields, a specification of parameters for the emitter and a function that performs a Map operation as it is showed in Listing 1. A Reduce operation is implemented in Java. A workflow user chooses its class in a workflow description. In Fig. 1b shows a graphical representation of discussed approach.

5. Results

In a purpose of proving good abilities of defining Map/Reduce operations in WS-VLAM, the word count application was implemented. Map operation splits a text into words and for each word it emits a pair `(<word>, 1)` where `<word>` stands for one word from the text. Its implementation is presented in Listing 1. Then, in a Reduce operation these records are grouped basing on `<word>` and all values are added. The sum stands for a total number of occurrences of a particular word. The applicable implementation is already included into Hadoop and after adopting it to our needs, it can be reused. Tests were performed in DAS-4 environment that consists 8 nodes of dual quad-core 2.4 GHz CPUs and 24 GB memory each connected with InfiniBand and Gigabit Ethernet. We prepared 2.6 GB of English books in a plain text format taken from Project Gutenberg. They were stored in HDFS and used in tests as a whole set and smaller subsets of 2.6 GB, 1.3 GB and 930 MB of texts. The word count implementation in Java was executed and compared with elaborated solution. Execution command that runs our application is included in Listing 2. It uses implementation of the Map operation showed in Listing 1.

Table 1 gathers the execution times. It shows that slowdown is increasing with a grow of input sets. Nearly linear relation implies that a management and administration share in overall cost is decreasing while the actual computation cost increasingly influence a total execution time. It is hard to compare execution times of our test with the results achieved by the Kepler-based solution. Tests were performed in a different environments and on

Listing 2. Executing word counter implemented with MRToolkit (map operation presented in Listing 1). Line 3 specifies Map operation while 4 specifies Reduce. Line 5 contains a list of files that have to be available on working nodes – they are required by the `MapWords` class.

```
1 hadoop jar \$HADOOP_HOME/contrib/streaming/hadoop-streaming-1.0.0.jar \
2     -input texts -output output \
3     -mapper "ruby wordcount.rb -s MapWords" \
4     -reducer SumReducer \
5     -file wordcount.rb -file mrtoolkit/lib/stream_runner.rb -file mrtoolkit/lib/mrtoolkit.rb
```

Table 1. Word counter execution times.

Size of input data	Time (Java)	Time (map operation in MRToolkit)	Slowdown in %
2.6GB (6144 items)	626 s	1073 s	71%
1.3GB (3534 items)	367 s	620 s	69%
930MB (2102 items)	234 s	382 s	63%

different data. However, we gained better relative execution times – our solution was slower by 63-71% comparing to the Java implementation while the Kepler-based solution was 4-6 times slower [8].

6. Conclusions and future work

The elaborated approach provides a simple environment for defining MapReduce queries in workflow models. It follows the examples of Pig and Sawzall applications and their DSL-based solutions. The performance of our application is acceptable and metaprogramming approach is considered to be easy to use. However, more complex applications have to be investigated in the future. However, the presented solution has a one major disadvantage – information processed by Ruby (but also by the other executable run with Hadoop Streaming interface) in a Map operation, looses its type and then, it has to be cast to a required data type in Reduce phase.

In the future work, other programming languages can be considered as an alternative to Ruby, specially, statically typed languages based on Java Virtual Machine platform such as Scala programming language [12]. They can provide good constructs for metaprogramming approach and at the same time, they can be directly used in the Hadoop environment in the same JVM instance. It would let us to resign from Hadoop Streaming interface and to gain better performance and robustness.

Acknowledgements. This work was partially supported by the Dutch national program COMMIT. We would like to thank Reginald Cushing and Spiros Koulouzis from University of Amsterdam for discussions and suggestions.

References

[1] T. Hey, S. Tansley, K. Tolle (Eds.), The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, Redmond, Washington, 2009.
[2] S. Koulouzis, R. Cushing, K. Karasavvas, A. Belloum, M. Bubak, Enabling web services to consume and produce large datasets, Internet Computing, IEEE 16 (1) (Jan.-Feb.) 52–60.
[3] A. Belloum, M. Inda, D. Vasunin, V. Korkhov, Z. Zhao, H. Rauwerda, T. Breit, M. Bubak, L. Hertzberger, Collaborative e-science experiments and scientific workflows, Internet Computing, IEEE 15 (4) (July-Aug.) 39–47.
[4] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[5] A. S. Foundation, Apache hadoop, `http://hadoop.apache.org/`.
[6] C. Olston, B. Reed, U. Srivastava, R. Kumar, 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, SIGMOD '08, ACM, New York, NY, USA, 2008, pp. 1099–1110.
[7] R. Pike, S. Dorward, R. Griesemer, S. Quinlan, Interpreting the data: Parallel analysis with sawzall, Scientific Programming 13 (4) (2005) 277.
[8] J. Wang, D. Crawl, I. Altintas, Kepler + hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems, in: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, WORKS '09, ACM, New York, NY, USA, 2009, pp. 12:1–12:8.
[9] C. Stefansen, Smawl: A small workflow language based on ccs, in: Harvard University, 2005.
[10] M. Baranowski, A. Belloum, M. Bubak, M. Malawski, Constructing workflows from script applications, Scientific Programming.
[11] N. Y. Times, Mrtoolkit, `http://code.google.com/p/mrtoolkit/`.
[12] P. M. L. of EPFL, Scala programming language, `http://www.scala-lang.org/`.