BDA: A Platform for Big Data Analysis

[Extended Abstract] *

Ben Trovato†
Institute for Clarity in Documentation
1932 Wallamaloo Lane
Wallamaloo, New Zealand
trovato@corporation.com

G.K.M. Tobin‡
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

Lars Thørväld§
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

Lawrence P. Leipuner
Brookhaven Laboratories
Brookhaven National Lab
P.O. Box 5000
lleipuner@researchlabs.org

Sean Fogarty
NASA Ames Research Center
Moffett Field
California 94035
fogartys@amesres.org

Charles Palmer
Palmer Research Laboratories
8600 Datapoint Drive
San Antonio, Texas 78229
cpalmer@prl.com

ABSTRACT

Big data processing and machine learning have been made the world a better place. However, configuring machine learning task is always a complicated job. BDA is a GUI-based platform for big data analysis and mining, which provides a wealth of machine learning algorithms. Based on Hadoop and Spark, BDA is able to process massive datasets. One begins a task by construct dataflow based acyclic graph of algorithms with GUI. Each link in the acyclic graph represents a dataflow between algorithms. Machine learning is made easy, beautiful and understandable in BDA.

CCS Concepts

•**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Keywords

Data Mining; Big Data Analysis; Machine Learning

1. INTRODUCTION

Data mining and analysis have been widely use in many industry verticals, such as financial services, communications, retail and e-commerce. However, configuring and going through the whole roadmap of data mining and analysis will be a complicated job. Integrated applications that facilitate going through the data mining and analysis job are widely demanded for data scientists and business analysts.

Traditional tools like SAS, SPSS show their strength on descriptive and diagnostic analytics, and are useful for static structural datasets. However, they are not cloud-based, which limited the usage of advanced techniques in distributed systems. Systems like Azure Machine Learning and Alibaba Yushanfang are modern cloud platform for data scientists to design data experiments.

Modern analysis tools shares features, such as native support advanced analytic modeling techniques, visualization of dataflow design, ability to save dataflows into files and libraries for later reuse, allowing program using scripting language.

We present a modern cloud-based Big Data Analysis(BDA) system, which meets features described above. It's design on distributed system, Hadoop and Spark, and choose Oozie as job scheduler.

2. TECHNOLOGY SPECIFICATION

2.1 System Architecture

BDA is a GUI-based platform designed on distributed system. It applied HDFS as storage, Spark and Map-Reduce as computation framework, and Oozie as job scheduler. Figure 1 shows system architecture.

We divide the architecture into three hierarchy: Resource Management Layer, Schedule and Computation Layer and User Interface Layer.

- **Resource Management Layer:** This layer focuses on how to store the resources and datasets. We use database to store meta data of programs and jobs, while programs were stored on HDFS. All resources are

*A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using $\LaTeX2_\epsilon$ and BibTeX* at `www.acm.org/eaddress.htm`

†Dr. Trovato insisted his name be first.

‡The secretary disavows any knowledge of this author's actions.

§This author is the one who did all the really hard work.

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

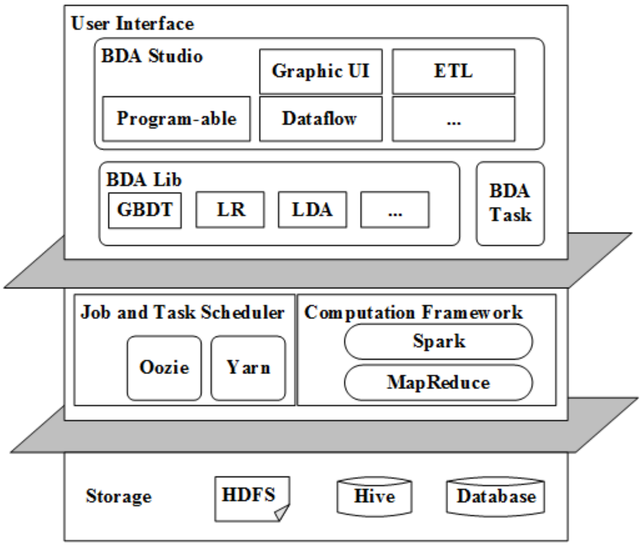

Figure 1: A overview of BDA architecture.

stored on HDFS. And, Hive was applied to facilitates reading, writing, and managing large datasets residing in HDFS using SQL.

- **Schedule and Computation Layer:** This layer focuses on how bda jobs are dispatched and executed. BDA applied Oozie as job scheduler. Each actions in the job will be dispatched to any node in the Haddop cluster by YARN and run as a Map task. Spark and Map-Reduce is our foundation of distributed computation framework.
- **User Interface Layer:** This layer focuses how to provide friendly functionality for data scientists. There are three components, BDA Studio, BDA Lib and BDA Task. BDA Studio is a web service which support graphic-based components, program-able components, and dataflow based acyclic job graph constructions. BDA Lib provides a wealth of machine learning algorithms. BDA Task provides examples of real use cases.

We will introduce how these layer cooperation.

3. BDA LIB

BDA Lib is the algorithm library of BDA, it provides a wealth of machine learning, data mining algorithms. We have both Standalone and Spark implementations for each algorithm. All these algorithms are uploaded to BDA system and categorized as pre-processing, transformation, machine learning, and evaluation. Besides, we provide a sort of API for BDA Lib.

When uploading to BDA, the usage command line format of algorithm must be correctly configured. Otherwise, algorithm won't run successfully. The real command line would be generated when a BDA job is submitted, with configurations of algorithms. The command line format must be defined following the rules bellow:

- **For Input File Position Holder:**
  {in:ContentType:"descriptions"}
- **For Output File Position Holder:**
  {out:ContentType,StorageType:"descriptions"}
- **For Numerical Parameter Position Holder:**
  ["parameter name":Type,min,max:default,value]
- **For Boolean Parameter Position Holder:**
  ["parameter name":Bool:default,value]
- **For String Parameter Position Holder:**
  ["parameter name":String:default,"value"]

ContentType specifies the format of file contents, where General as general files, LibSVM, TSV, CSV, Binary, JSON, XML.

StoreType specifies how the resources is store on HDFS, where SFile as a single file, HFile as a Hadoop distributed file, and Directory.

For numerical parameter, Type specifies the parameter types which may be Integer, Double or Float. In additions, the min and max value could be skipped if there are no minimum or maximum limitations.

Here comes an example of logistic regression(LR) command line format, it specifies the usage of LR train. *train_pt*, *validate_pt*, and *model_pt* are input files. *optimizer*, *max_iter*, *reg* and *learn_rate* are parameters of LR.

$$
\begin{array}{ll}
java\ -cp\ local.jar & bda.local.runnable.LR.Train \\
--train\_pt & \{in : LibSVM : "train\ set"\} \\
--validate\_pt & \{in : LibSVM : "validate\ set"\} \\
--model\_pt & \{out : Binary, HFile : "model"\} \\
--optimizer & ["opt" : String : default, "sgd"] \\
--max\_iter & ["max\_iter" : Integer : default, 20] \\
--reg & ["reg" : Double : default, 0.01] \\
--learn\_rate & ["learn\_rate" : Double : default, 0.1]
\end{array}
$$

4. BDA STUDIO

BDA Studio is a GUI-based web service on the top of BDA architecture, allowed data scientists to access to resources and services provided by BDA easily. BDA Studio provides services like job schedule, distributed dispatch and execution, account management and resource management.

Figure 2 shows the process to perform a job. As is shown, one must construct a dataflow based acyclic graph of algorithms and datasets, where each arrow specify a dataflow. When a well-formed job graph is submitted, it will be scheduled by Oozie, and finally run on the cloud. We also provide job clone, rerun and edit functionalities which facilitates the reuse of job schemas after the job is finished.

The main technical challenges we have encountered are: GUI based DAG construction; dataflow based on Oozie workflow; enable program running on the cloud; ETL.

Besides, user custom programs are allowed to submit to the platform with configuration. In addition, BDA also provide program-able components, which support SQL, Spark, and Shell.

4.1 Dataflow Based Acyclic Graph

We introduce dataflow based acyclic graph into BDA system based on Oozie workflow. And Oozie is a workflow scheduler system, where workflow is a collection of actions arranged in a control dependency DAG(Directed Acyclic

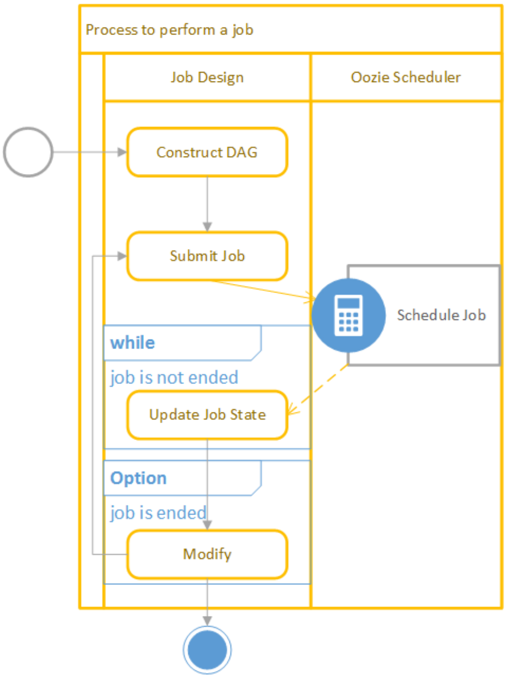

Figure 2: Process to perform job.

Graph). Control dependency from one action to another means that the second action can't run until the first action has completed. The concept dataflow differed with workflow as to dataflow specify data dependency between different actions. And the control dependency of actions are determined by the data dependency.

Figure 3 is an example for dataflow based acyclic job graph. Each algorithm or dataset submitted to BDA, will be represented as a GUI component in BDA Studio, where the points on the top of component are dataflow input anchor points and on the bottom are dataflow output anchor points. Anchor points are generated according to algorithm data parameters. Each arrow represent as a data dependency between components. In addition, the arrow's start and end anchor points must be matched, otherwise the algorithm will not run correctly.

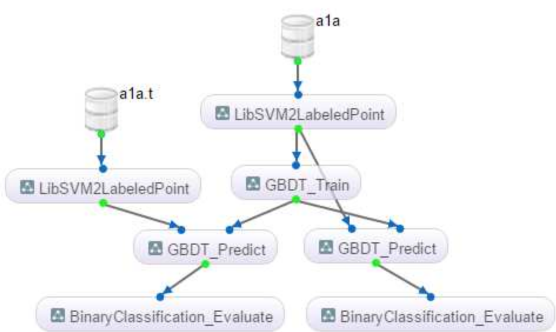

Figure 3: An example for dataflow based acyclic job graph.

4.2 Program-able Component

WhatâĂŹs program-able component? ItâĂŹs important to ETL. The significance of ETL. The technology specification

5. APPENDIX

6. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LaTeX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

7. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TeX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LaTeX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.6 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY

The sig-alternate.cls file itself is chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LaTeX, you may find reading it useful but please remember not to change it.