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 Directed Acyclic Graph (DAG) of algorithms with GUI. Each link in the DAG represents a data flow 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

*A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2ε 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

With the great advances in big data processing and machine learning, data mining application is close to daily life than ever before. For examples, commercial recommendation systems applied a lot of big data processing and machine learning techniques, in order to provide more accurate recommendations. A classical roadmap of creating recommendation task shown as follow: data collecting; data extraction, transform and load (ETL); feature engineering; model selection and evaluation; online application. While, configuring and going through all these process is a complicated job. An integrated platform is needed to enable one begin a machine learning task easily.

Analytic applications have been widely use in many industry verticals, such as financial services, communications, retail and e-commerce. 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. Besides, traditional tools has deficiency in predictive and prescriptive analytics of massive non-structural data. Systems like Azure Machine Learning and Alibaba Yushanfang are modern cloud platform for data scientists to design data experiments. They are very scalable and easy use. Also, they have advantages in big data processing.

We present a modern cloud-based Big Data Analysis(BDA) system. The system consists of three components: BDA Studio, BDA Lib, BDA Task. It's a Web service based on Hadoop and Spark. In addition, we develop a dataflow framework based on Oozie workflow.

Our BDA, objective, advantages, main features. The papers framework.

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. Fig-

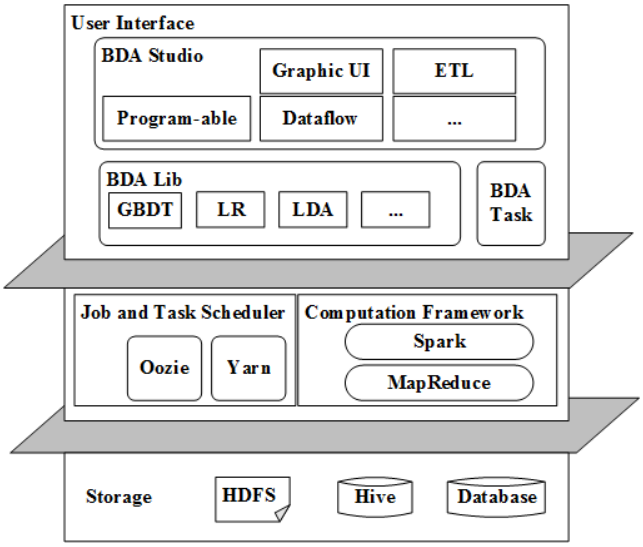

Figure 1: A overview of BDA architecture.

ure 1 shows system architecture.

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

- **Storage 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 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 data flow based job DAG 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 STUDIO

BDA Studio is a GUI based web service, which allows data scientist to construct an analytic tasks by drawing DAGs of algorithms and datasets. When a well-formed job DAG is submit, it will be scheduled by Oozie, and finally run on the cloud. We provide job clone, rerun and edit functionalities which facilitates the reuse of job schemas. 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.

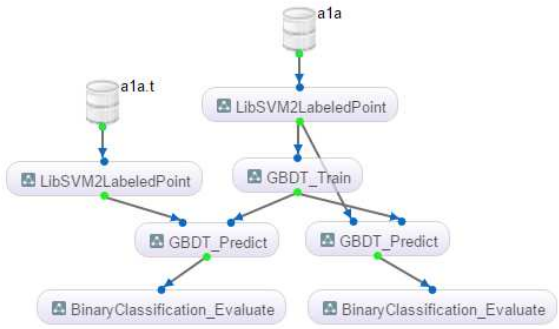

Figure 2: An example of job DAG.

The main technical challenges we encountered are: how to support GUI based DAG construction; how to generate workflow configuration that will be submit to Oozie; how to enable program running on the cloud; how to support ETL.

3.1 GUI based Algorithm Components

Each algorithm or dataset submitted to BDA, will be represented as a GUI component in BDA Studio. One could click to add a component to a BDA job. In additions, the data dependencies between different components are presented as arrows. Figure 2 shows an example of job DAG.

3.2 Dataflow

As shown in Figure 2, BDA job is dataflow based. Which differs from Oozie workflow. A workflow defined how processes are scheduled, but the data dependency is not specified. In order to support dataflow job description, we first collect the data dependencies of different components in the BDA job DAG, then solve the components dependencies which is a really a workflow.

3.3 Command Line Format

A program without usage guidance couldn't be run correctly. In BDA, the usage command line format must be specified when a program is uploading. The real command line would be generated when a BDA job is submitted, with the structure of DAG and parameter configurations of components. 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"]

Here comes an example of logistic regression(LR) command line format, it specifies the usage of LR. $train\_pt$, $validate\_pt$, and $model\_pt$ are input files. $optimizer$, max_iter, reg and $learn\_rate$ are parameters of LR.

$java\ -cp\ local.jar$	$bda.local.runnable.LR.Train$
$--train\_pt$	$\{in : LabeledPoint : "train\ set"\}$
$--validate\_pt$	$\{in : LabeledPoint : "validate\ set"\}$
$--model\_pt$	$\{out : LRModel, HFile : "model"\}$
$--optimizer$	$["opt" : String : default, "sgd"]$
$--max\_iter$	$["max\_iter" : Int : default, 20]$
$--reg$	$["reg" : Double : default, 0.01]$
$--learn\_rate$	$["learn\_rate" : Double : default, 0.1]$

What is BDA job? Based on oozie, and different from it. (oozie is a workflow system, while BDA is dataflow) BDA job: clone-able, editable, result-reuse

3.4 Program-able Component

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

4. APPENDIX

5. 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.

6. 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 `\addition-alauthors` 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.