Symbolic Execution of Apache Spark Programs

Omar A. Erminy Ugueto
Fachbereich Informatik
February 20, 2017

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Omar Erminy
Matriculation Number: 2996125
Study Program: Master in Distributed Software Systems

Master Thesis
Topic: Symbolic Execution of Apache Spark Programs

Submitted: February 20, 2017

Supervisor: Prof. Dr. Guido Salvaneschi

Prof. Dr-Ing. Mira Mezini
Fachgebiet Softwaretechnik
Fachbereich Informatik
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt

Abstract

Informationen zu Inhalten der Zusammenfassung entnehmen Sie bitte Kapitel 6.1 des Skripts zur Veranstaltung *Wissenschaftliches Arbeiten und Schreiben für Maschinenbau-Studierende*.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction

2 Related Work

2.1 Apache Spark

Spark is a data processing framework that was first introduced in 2012 [1]. Similar to other systems, such as MapReduce [2] and Dryad [3], it aims to offer a clean and flexible abstraction to distributed computations on large datasets. However, Spark offers two advantages in comparison to such systems: It makes use of a shared memory abstraction that improves performance by avoiding persisting intermediate sets. It also provides an efficient fault-tolerance mechanism, based on tracking coarse-grained operations, that can recover lost tasks with minimal impact.

The working units in Spark are called *Resilient Distributed Datasets*, better known as RDDs. These units represent an immutable partitioned collection of elements in a distributed memory space. RDDs can only be created through a set of deterministic operations, known as *transformations* (e.g., *map*, *filter* and *join*), that can be applied to both, raw data or other RDDs. Transformations are not evaluated immediately, instead Spark keeps track of all the transformations applied to each RDD in a program so it can optimize their subsequent processing. Additionally, RDDs can be made persistent into storage or can be operated to produce a value. This kind of operations are known as *actions* (e.g., *count*, *reduce* and *save*), and they are the ones that trigger the processing of RDDs.

To interact with the RDD abstraction, Spark provides several APIs for different programming languages such as Java, Scala, Python and recently R [4]. Listing 2.1 presents a simple Spark program written with the Scala API, that processes log files in the search for errors. The operation in line 1 creates the first RDD from a log file, whose origin could be a local file or a partitioned file in a distributed file system such a Hadoop Distributed File System (HDFS) [5]. Spark converts each line in the file to a *String* element in the newly created RDD. In lines 2 to 4, a chain of transformations is applied to the RDD: First, elements not containing the text “ERROR” are filtered. Next, each resulting element is transformed to a tuple consisting of a certain property (e.g., error type; assumed to be the first information in a log entry) and the number 1. Finally, the tuples are grouped and counted based on the chosen property. Line 5 represents the action applied to the RDD, in this case, saving it to persistent storage.

```
1 val log = spark.textFile("*file*")
2 val errors = log.filter(_.contains("ERROR"))
3   .map(error => (error.split('\t')(0),1))
4   .reduceByKey(_+_)
5 errors.save()
```

Listing 2.1: Entries in a log file are filtered, grouped and counted based on a common property. Finally the result is saved to persistent storage.

During the execution of a program, Spark does not generate imperatively new data collections for every transformation it finds. Instead, it constructs new RDDs attached with the operation that has to be applied to each element. The resulting RDD is a sequence of operations starting from the source dataset, whose semantics depends on the nature of each transformation applied. It is not until an action is found that the target RDD is resolved and the whole sequence of transformations actually operates the data.

Delaying the resolution of RDDs in this way allows Spark to improve the distribution of operations in a clustered dataset, taking advantage of properties like data locality. Moreover, the trace of operations that produced a certain element in an RDD, known as *lineage*, enables Spark to recover failed tasks only recalling to the necessary data elements that reproduce the lost portion. Figure 2.1 depicts the resulting lineage of the program explained in listing 2.1.

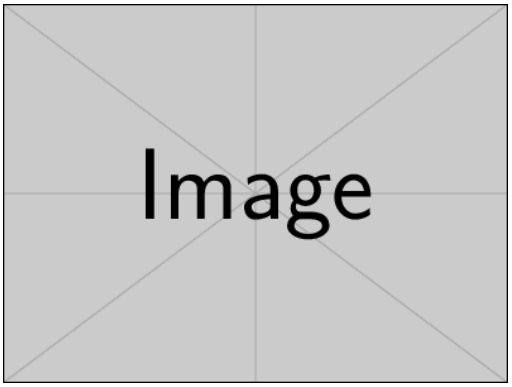

Figure 2.1: Lineage of a simple Spark program.

Most of the operations in Spark are higher-order functions, this means they accept one or more functions as parameters. For example, the *filter* transformation requires a function that takes an element of the RDD and evaluates to a boolean value. These user-defined functions work as closures by scoping their environment even if it contains references to variables outside it; this allows Spark to ensure consistency when applying such functions in parallel nodes. The use of higher-order functions provides a flexible mechanism to adapt Spark's computation model to different tasks.

The inherent capacity of Spark to operate in a distributed memory space, makes it well-suited for two particular scenarios: iterative algorithms and interactive querying. The former, which are commonplace among machine learning algorithms, leverages on the reuse of datasets and avoids having to perform costly I/O operations for every iteration. The latter, allows data mining techniques to synthesize queries faster by keeping working data at hand.

Spark is part of the Apache Software Foundation and it is offered as an open-source software [6, 7]. Several purpose-specific libraries are built on top of Spark, as is the case of: MLlib for machine learning [8], GraphX for graph computations [9], Spark Streaming for stream processing [10], and Spark SQL, an SQL-like interface for structured querying in Spark [11].

In 2014, Spark reported the fastest Daytona GraySort as defined by the Sort Benchmark committee, and later in 2016, Spark was part of the technology stack that claimed the most resource-efficient Daytona CloudSort as defined by the same committee [12, 13, 14]. Overall, Spark offers a better performance in comparison to other data processing frameworks.

2.2 Symbolic Execution

2.3 Java Path Finder (JPF)

3 Evaluation

4 Future Work

5 Einrichtung und Erläuterungen

Es wird empfohlen, bei der Installation einer LaTeX-Distribution die Optionen *automatische Updates* bzw. *on the fly-Installation* zu aktivieren. Dies hat den Vorteil, dass bislang nicht installierte Pakete bei Bedarf automatisch nachinstalliert werden.

Um das modifizierte TU-Design zu verwenden, müssen Sie zunächst das offizielle TU-Design installieren. Die dazugehörigen, betriebssystemabhängigen Installationsanleitungen finden Sie unten.
In Ihrem Dokument laden Sie dann das Paket *tustyle*. Standardmäßig ist die Layoutfarbe rot, sie können diese jedoch nach Belieben ändern. Hierzu laden sie das Paket mit der Option Ihrer gewünschten Farbe. Sollte Ihre Farbe noch nicht in der Klasse vorgesehen sein, können Sie diese selbstständig hinzufügen. Orientieren Sie sich hierbei an den Zeilen 144 bis 150 in dem Paket. Um eine neue Farbe zu definieren, benötigen Sie lediglich ihren RGB-Code.

Beispiel: \usepackage[blue]{tustyle} lädt das Paket *tustyle* mit der Farbe *blau*

Sollten Sie die offiziellen TU-Klassen verwenden, können Sie die Layoutfarbe als Option der Dokumentenklasse übergeben. Die möglichen Farben entnehmen Sie bitte der Handbuch des Corporate Designs.

Beispiel: \documentclass[accentcolor=tud9c]{tudreport} lädt die Dokumentenklasse tudreport mit dem Farbe 9c (weinrot, Farbe dieses Dokuments)

5.1 Einrichtung unter Windows

5.1.1 Setup

Zu Beginn wird für den Start folgendes Setup vorgeschlagen:

- LaTeX-Distribution: MiKTeX [Zugriff: 13.12.2014]
- LaTeX-Editor: TeXstudio [Zugriff: 13.12.2014]
- Literaturverwaltung (optional): Endnote/Citavi [Zugriff: 13.12.2014] mit TU-Lizenz der ULB oder das kostenfreie JabRef [Zugriff: 13.12.2014]
- PDF-Reader (optional): Sumatra PDF [Zugriff: 13.12.2014] (Adobe Reader verhindert den Kompiliervorgang bei geöffnetem PDF-Dokument)

Es gibt eine Vielzahl kostenloser und kostenpflichtiger LaTeX Distributionen (Auswahl [Zugriff: 13.12.2014]) und Editoren (Übersicht [Zugriff: 13.12.2014]) mit unterschiedlichem Funktionsumfang, mit denen persönliche Vorlieben erfüllt werden können. Aus Gründen der Einfachheit und Reproduzierbarkeit beziehen sich Hilfestellungen sowie Tipps und Tricks dieser Einrichtungshilfe allerdings auf oben genanntes Setup. Die Installation der genannten Komponenten ist unproblematisch und sollte mit den jeweiligen Installationsprogrammen durchgeführt werden können.

Für eine problemlose Kompilierung unter Windows 7 wird empfohlen, die Miktexversion für 32 Bit zu verwenden.

5.1.2 Installation der TU-Design-Vorlage für LaTeX

Die Vorlage für Abschlussarbeiten greift für die Umsetzung des Corporate Designs der TU Darmstadt auf die TUD-Design LaTeX Vorlage [Zugriff: 13.12.2014] zurück, welche von der Stabstelle Kommunikation und Medien genehmigt wurde. Diese hält die Vorgaben des Corporate Design Handbuchs (CDH) recht strikt ein (strikter als viele Fachgebiete dies bei den jeweils eigenen Word-Vorlagen tun), weshalb manche Anpassungen an Institutsvorgaben u.U. nur schwer umsetzbar sind, da sie gegen das CDH verstoßen.
Die notwendigen Pakete für die Verwendung der Vorlage für Abschlussarbeiten sind

- das TUD-Design [Zugriff: 13.12.2014]
- die TUD- Fonts [Zugriff: 13.12.2014]

Hinweis: die TUD-Design Thesis Klasse, welche ebenfalls zum Download bereit steht, wird nicht benötigt.

Installation

Für die Installation der TUD-Design Vorlage unter der MiKTeX Distribution unter Windows 7 kann folgende überarbeitete Anleitung verwendet werden. Sie basiert auf der Installationsanleitung auf den Seiten der TUD-Design Vorlage [Zugriff: 13.12.2014].

Hinweis: Für die Installation werden Administrator-Rechte benötigt

1. Entpacken der beiden Zip-Dateien (fonts und tuddesign) und anschließend aus den beiden Ordnern einen machen (ineinander kopieren und Verzeichnisse überschreiben)

2. Öffnen der Eingabeaufforderung mit Administratorrechten: Start » Programme » Zubehör, dann Rechtsklick auf Eingabeaufforderung » „Als Administrator ausführen“

3. Mit `cd <Pfad>` in das Verzeichnis wechseln, in dem der in 1 angelegte Ordner `texmf` liegt. Falls der texmf-Ordner auf einem anderen Laufwerk als Cliegt, muss beim Verzeichniswechsel der Parameter `/d` angegeben werden:
   Beispiel:
   texmf-Ordner liegt unter `E:\Test`
   Befehl: `cd /d E:\Test`

4. Löschen des Ordners `texmf\fonts\map\dvipdfm` inklusive seines Inhalts mit folgendem Befehl
   `rmdir /Q /S "C:\Users\Benutzername\TU-Design\texmf\fonts\map\dvipdfm"`

5. Kopieren der Unterverzeichnisse von texmf in den Ordner `\%PROGRAMFILES%\tuddesign\` mit folgendem Befehl:
   `xcopy texmf "%PROGRAMFILES%\tuddesign" /E /I`
   Falls der xcopy Befehl fehlschlägt, liegen keine Administratorrechte vor (keine Schreibrechte für Programme-Ordner)

6. Dannach folgenden Befehl ausführen:
   `mo_admin`
   Zum Reiter Roots wechseln, den *add*-Knopf drücken und das Verzeichnis `\%PROGRAMFILES%\tuddesign` auswählen. (Unterordner tuddesign im Standard-Programmverzeichnis der Windows-Partition - i.d.R. auf `C:\`)
   Dann auf *OK* klicken.

7. In der Konsole folgendes eingeben:
   `initexmf --admin --update-fndb`

8. Anschließend Folgendes eingeben
   `initexmf --edit-config-file=updmap`

9. Folgende Zeilen in die sich öffnende Datei einfügen und speichern:

Map 5ch.map
Map 5fp.map
Map 5sf.map

10. Abschließend diesen Befehl ausführen
`initexmf --mkmaps`

Bibliography

[1] Zaharia, M. et al. “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing”. In: *NSDI’12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (2012), pp. 2–2. ISSN: 00221112. DOI: 10.1111/j.1095-8649.2005.00662.x. arXiv: EECS-2011-82.

[2] Dean, J. and Ghemawat, S. “MapReduce: Simplied Data Processing on Large Clusters”. In: *Proceedings of 6th Symposium on Operating Systems Design and Implementation* (2004), pp. 137–149. ISSN: 00010782. DOI: 10.1145/1327452.1327492. arXiv: 10.1.1.163.5292.

[3] Isard, M. et al. “Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks”. In: *ACM SIGOPS Operating Systems Review* (2007), pp. 59–72. ISSN: 01635980. DOI: 10.1145/1272998.1273005.

[4] Venkataraman, S. et al. “SparkR: Scaling R Programs with Spark”. In: *Sigmod* (2016), p. 4. ISSN: 07308078. DOI: 10.1145/1235. arXiv: arXiv:1508.06655v1.

[5] *Welcome to Apache™ Hadoop®!* URL: http://hadoop.apache.org/ (visited on 2017).

[6] *Welcome to The Apache Software Foundation!* URL: https://www.apache.org/ (visited on 2017).

[7] *Apache Spark™ - Lightning-Fast Cluster Computing*. URL: http://spark.apache.org/ (visited on 2017).

[8] Meng, X. et al. “MLlib : Machine Learning in Apache Spark”. In: *Journal of Machine Learning Research* 17 (2016), pp. 1–7. arXiv: arXiv:1505.06807v1.

[9] Xin, R. S. et al. “GraphX: A Resilient Distributed Graph System on Spark”. In: *First International Workshop on Graph Data Management Experiences and Systems - GRADES ’13* (2013), pp. 1–6. ISSN: 0002-9513. DOI: 10.1145/2484425.2484427. arXiv: 1402.2394.

[10] Zaharia, M. et al. “Discretized Streams: Fault-Tolerant Streaming Computation at Scale”. In: *Sosp* 1 (2013), pp. 423–438. DOI: 10.1145/2517349.2522737.

[11] Armbrust, M. et al. “Scaling spark in the real world: performance and usability”. In: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 1840–1843. ISSN: 21508097. DOI: 10.14778/2824032.2824080.

[12] *Sort Benchmark Home Page*. URL: http://sortbenchmark.org/ (visited on 2017).

[13] Xin, R. et al. *GraySort on Apache Spark by Databricks*. Tech. rep. 2014.

[14] Wang, Q. et al. *NADSort*. Tech. rep., pp. 1–6.

6 Declaration of Academic Integrity

Thesis Statement pursuant to § 22 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

In the submitted thesis the written copies and the electronic version are identical in content.

Date: Signature:

List of Figures

List of Tables

List of Listings

Glossary

Lineage	Lineage description

List of Abbreviations and Acronyms

API	Application Programming Interface
HDFS	Hadoop Distributed File System
RDD	Resilient Distributed Dataset

A Anhang

A.1 Ein Anhang

Hier gibt es etwas zu sagen oder auch nicht.

A.1.1 Teil eines Anhangs

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

A.1.2 Noch ein Teil eines Anhangs

- First item in a list
- Second item in a list
- Third item in a list
- Fourth item in a list
- Fifth item in a list

A.2 Noch ein Anhang

Hier gibt es etwas zu sagen oder auch nicht.

A.2.1 Teil des weiteren Anhangs

1. First item in a list
2. Second item in a list
3. Third item in a list
4. Fourth item in a list
5. Fifth item in a list

A.2.2 Noch ein Teil des weiteren Anhangs

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{i=n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n} \tag{A.1}$$

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

$$\int_0^\infty e^{-\mathrm{ax}^2}\mathrm{d}x = \frac{1}{2}\sqrt{\int_{-\infty}^{\infty} e^{-\mathrm{ax}^2}\mathrm{d}x \cdot \int_{-\infty}^{\infty} e^{-\mathrm{ay}^2}\mathrm{d}y} = \frac{1}{2}\sqrt{\frac{\pi}{\alpha}} \tag{A.2}$$

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original

language. There is no need for special content, but the length of words should match the language.

$$\sum_{k=0}^{\infty} a_0 g^k = \lim_{n\to\infty} \sum_{k=0}^{n} a_0 q^k = \lim_{n\to\infty} a_0 \frac{1-q^{n+1}}{1-q} = \frac{a_0}{1-q} \tag{A.3}$$

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-p \pm \sqrt{p^2 - 4q}}{2} \tag{A.4}$$

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} = \frac{1}{c^2} \frac{\partial^2 \Phi}{\partial t^2} \tag{A.5}$$

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.