Symbolic Execution of Apache Spark Programs

Omar A. Erminy Ugueto
February 27, 2017
Fachbereich Informatik
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 27, 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.

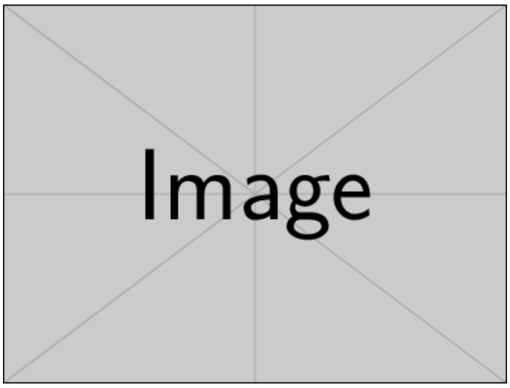

Figure 2.1: Lineage of a simple Spark program.

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 Formal Methods - Symbolic Execution

(Model Checking, why should we use it?)

(Related to Model Checking — First of all, programs often contain fatal errors despite the existence of careful designs.Many deadlocks and critical section violations, for example, are introduced at a level of detail which designs typically do not deal with, if formal designs are made at all.)

(Related to Model Checking — The other kind of error is more simple minded concurrency programming errors, such as forgetting to put code in a critical section or causing deadlocks. Errors of this kind will typically not be caught in a design, and they are a real hazard, in particular in safety critical systems.)

2.3 Java PathFinder

Developed at NASA's Ames Research Center [15], Java PathFinder (JPF) is an execution environment for verification and analysis of Java bytecode programs [16, 17]. Since its publication in the year 2000 [18], JPF has evolved from being a model translator to a fully fledged, highly customizable virtual machine capable of controlling and augmenting the execution of a program.

Java is a widely known, general-purpose programming language with strong roots on concurrency support and object-oriented principles [19]. Programs written in Java are compiled to the standardized instruction set of the Java Virtual Machine (JVM), known as Java bytecode. This process enables Java programs to be portable between architectures implementing the JVM specification. A JVM implementation serves as an interpreter of Java bytecode and allows the optimization and execution of the program tailored for the host platform [20].

JPF focused on Java mainly for three reasons: its wide adoption as a modern programming language, its simplicity in comparison to other high profile languages, and the flexibility in terms of bytecode analysis; potentially enabling the verification of any other language capable of being compiled into Java bytecode. Moreover, the non-trivial nature of concurrent programs makes them difficult to construct and debug. A model checker with the capacity of validating concurrent Java programs would have proven crucial for ensuring correctness of mission-critical software, such as the likes required by NASA.

In its core, JPF is a Java Virtual Machine implemented in Java itself, comprised of several extensible modules that dictate the verification strategy to be followed. The fact that JPF is written in Java means that it is executed on a canonical JVM; in other words, a JVM on top of a JVM. Figure 2.2 portrays the components that participate in a verification process using JPF. The program under test is loaded into JPF's core, where its instructions are executed one by one until an execution choice is found. At this point, JPF records the current state and attempts to resume execution, exploring all possible scenarios based on the choice criteria. To try different options, JPF backtracks to a recorded state where a certain choice was made, and keeps track of the execution paths already attempted to ensure exploring only new paths.

(A major design decision for JPF was to make it as modular and understandable to others as possible, but we sacrificed speed in the process—Spin is at least an order of magnitude faster than JPF.We believe this

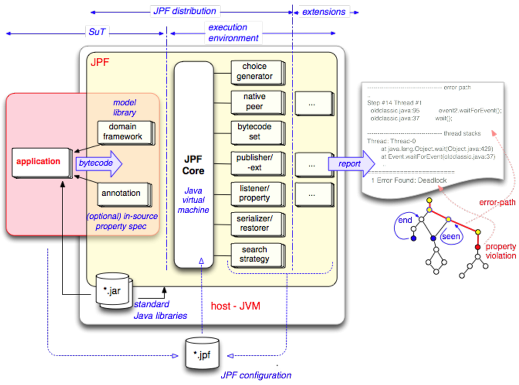

Figure 2.2: JPF Workflow

```java
import java.util.Random;

public class RandomExample {

  public static void main(String[] args) {
    Random random = new Random();
    int a = random.nextInt(2);
    int b = random.nextInt(3);
    int c = a/(b+a-2);
  }
}
```

Listing 2.2: The use of random values could lead to unexpected behavior. In this case, a division by zero could occur if certain combinations of random values are used.

is a price worth paying in the long run.)

(Termination — In order to ensure termination during explicit state model checking one must know when a state is revisited. It is common for a hashtable to be used to store states, which means an efficient hash function is required as well as fast state comparison.)

(States — Specifically, each state consists of three components: information for each thread in the Java program, the static variables (in classes) and the dynamic variables (in objects) in the system. The information for each thread consists of a stack of frames, one for each method called, whereas the static and dynamic information consists of information about the locks for the classes/objects and the fields in the classes/object)

(Features of JPF) (Explicit State, State Matching, Backtracking, Partial Order Reduction)

(Extension Points: Search strategies, Choice Generators, Listeners, Instruction Factories, Native Peers, Models, Publisher, consider presenting them as a list of concepts)

(Modules - Briefly mention them a an introduction to SPF)

2.3.1 Symbolic PathFinder (SPF)

(Definition of SPF)

(Explain its extension points: Choice Generators, Listeners, Symbolic Instruction Factory)

(Mention the solvers)

3 Evaluation

4 Future Work

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. 2016, pp. 1–6.

[15] *NASA's Ames Research Center*. National Aeronautics and Space Administration. URL: https://www.nasa.gov/centers/ames/home/index.html (visited on 2017).

[16] Visser, W. et al. "Model Checking Programs". In: (2003), pp. 203–232.

[17] *Java PathFinder*. National Aeronautics and Space Administration. URL: http://babelfish.arc.nasa.gov/trac/jpf/wiki (visited on 2017).

[18] Havelund, K. and Pressburger, T. “Model checking JAVA programs using JAVA PathFinder”. In: *International Journal on Software Tools for Technology Transfer (STTT)* 2.4 (2000), pp. 366–381. ISSN: 14332779. DOI: 10.1007/s100090050043.

[19] Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, A. “The Java® Language Specification - jls8.pdf”. In: *Addison-Wesley* (2014), p. 688.

[20] Lindholm, T. et al. “The Java® Virtual Machine Specification”. In: *Managing* (2014), pp. 1–626.

5 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 Installation of the JPF and everything else

A.2 Ein Anhang

Hier gibt es etwas zu sagen oder auch nicht.

A.2.1 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.2.2 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.3 Noch ein Anhang

Hier gibt es etwas zu sagen oder auch nicht.

A.3.1 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.3.2 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.