Boa: Ultra-Large-Scale Software Repository and Source Code Mining

By Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N. Nguyen
ACM Transactions on Software Engineering and Methodology (TOSEM) 2015

Delvison Castillo

Introduction

Software repositories such as SourceForge, Google Code, and GitHub contain an enormous corpus of software and related information. Scientists and engineers are interested in analyzing this wealth of information.

Problems

Very large-scale (if no compromises are made). Too much data!
There is no organized dedicated method for doing this.
Too much effort!

Why Mine Software Repo’s and Source Code?

(Curiosity) - Identify trends in Software Engineering.

(Testing) - Mining Software Repository (MSR) Hypothesis (such as..):
“How many projects continue to use DES encryption standards?”
“What is the average time to resolve a bug reported as critical?”
“How many projects on average start with an existing code base from another project as opposed from scratch?”

Problems with MSR Hypothesis Testing

Testing Mining Software Repository (MSR) Hypothesis usually requires:

Expertise in accessing version control systems (Ex: SVN, Git)

Establishment of an infrastructure for downloading and storing the data from software repositories.

Programming an analysis program in a full-fledged language like C++ or Python to access the local data and answer the hypothesis.

Improving the scalability of analysis infrastructure to be able to process ultra large scale data in a reasonable time.
Problems with MSR Hypothesis Testing (Cont’d)

Additional MSR Hypothesis Testing Issues:

Experiments are often times not replicable due to large effort required.

Reusability of experimental infrastructure is typically low because the analysis infrastructure is not usually designed to be reusable.

Data associated and produced by these experiments are often times lost since there is no systematic curation.

Building an analysis infrastructure to process ultra large scale data efficiently can be very hard.

Motivation for Boa

Consider this question:

“What are the average numbers of changed files per revision (churn rates) for all Java projects that use SVN?”

To begin to answer this question we need:

To read project metadata
Mine the code repository locations
Knowledge of how to access these code repositories
To additionally filter code

Assumes downloaded project metadata (JSON)
External JSON lib needed to parse metadata
Need to collect list of SVN URL’s
External SVN lib needed to connect to SVN URL’s and calculate the churn rate.

Sequential code doesn’t scale well
What if all of this were standardized?

Introducing Boa…

What is Boa?

Domain-specific programming language for analyzing ultra large scale software repositories.
Strongly influenced by the language Sawzall.
Runs on a dedicated cluster and utilizes a web application as its interface.
Users submit their Boa jobs via web interface.

What does Boa aim to do?

Accomplish ease of use
Substantially improve in scalability
A glimpse of Boa

Same Question: “What are the average numbers of changed files per revision (churn rates) for all Java projects that use SVN?”

Compared to the Java solution, Boa:

- Runs on a distributed cluster which saves much more time. (116 cores)
- Requires no external libraries or knowledge of how to find/access project metadata.

Abstracted from the user into a Project type (line 2).

Semantic model of Boa

Each project becomes one input to a process.

Program instantiated once per input (project).

Each instantiation will process a single project (calculate churn rate).

Overview of the Boa Infrastructure

1. Translate analysis question into Boa program
2. Feed program to compiler via web-based interface
   a. Boa compiler produces a query plan
3. Deploy query plan onto a Hadoop cluster to execute.
   a. The cluster makes use of a locally cached copy of the source code repositories which provides monthly snapshots.
   b. Based on the query plans, tasks to produce the results are decided.
Features of the Boa Language

- Domain Specific Types.
- Declarative visitors.
- MapReduce support.
- Quantifiers.
- Ability to define functions.

Boa's Domain Specific Types

- Source code is represented as Abstract Syntax Tree (AST).
- Flexible to support complex language features and new source code languages.

Examples:
- `foreach (String s : iter) { ... }`

Boa's Declarative Visitors

- Defines what nodes to traverse.
- Purpose: to ease source code mining.

Examples:
- Filter traversals to leave out unnecessary nodes.
- Custom visitors can specify what snapshot to visit.
Boa's Declarative Visitors (Cont’d)

Consider the MSR hypothesis:

“A large number of bug fixes add checks for null”

Declarative visitors would allow us to specify snapshots.

Before and after a bug is fixed.

Allows us to easily compare the nodes in the traversal of the AST.

MapReduce Support in Boa

MapReduce frameworks consist of:

Mapper - takes key-value pairs and aggregates data based on individual keys.

Reducer - consumes key-value pairs and aggregates data based on individual keys.

In Boa:

Users write the mapper functions directly and use built-in aggregators as the reduce function.

Examples of built-in aggregators:

- Sum
- Max

Quantifiers in Boa

Syntactic sugar to accommodate frequent task in mining

Facilitates programming and comprehension

Examples:

- foreach
- exists
- ifall

User-defined Functions in Boa

Built-in functions might not accommodate all uses.

User-defined functions offer:

- More control
- Reusability
- Compartmentalization and compactness
Evaluation of Boa

In almost every case, Boa is much faster and requires less LOC for MSR tasks than Java.

Evaluation of Boa (Cont’d)

Extremely compact and concise!

Does is scale well with input?

Boa shows drastic improvements in scalability without requiring the programmer to explicitly parallelize code.

Boa reduces programming efforts in Mining Software Repositories.

As a result, the reduced difficulty lowers the barrier to entry into this realm of studies which promotes more research in MSR testing.

Conclusion
Summary

Mining Software Repository (MSR) Hypothesis Testing

Boa Features:

- Domain Specific Types.
- Declarative visitors.
- MapReduce support.
- Quantifiers.
- Ability to define functions.