Boa
A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories

Robert Dyer  Hoan Anh Nguyen  Hridesh Rajan  Tien N. Nguyen

{rdyer,hoan,hridesh,tien}@iastate.edu

Iowa State University
Why mine software repositories?

Learn from the past → Inform the future

- What is actually practiced
  - Keep doing what works
- Empirical validation
- To find better designs
- Spot (anti-)patterns
- Keep doing what works
Consider a task that answers "What is the average churn rate for Java projects on SourceForge?"

Note: churn rate is the average number of files changed per revision
Is Java project?

Yes → Has repository?

Yes → Access repository

mine project metadata

foreach project

Calculate project's churn rate

mine revision data

Calculate average churn rate
A solution in Java...

public class GetChurnRates {
    public static void main(String[] args) {
        new GetChurnRates().getRates(args[0]);
    }
    public void getRates(String cachePath) {
        for (File file : FileIO.readObjectFromFile(cachePath)) {
            String url = getSVNUrl(file);
            if (url != null && !url.isEmpty())
                System.out.println(url + "\" + getChurnRateForProject(url));
        }
    }

    private String getSVNUrl(File file) {
        String jsonTxt = "";
        ... // read the file contents into jsonTxt
        JSONObject json = null, jsonProj = null;
        ... // parse the contents of the project json
        if (!jsonProj.has("programming-languages")) return "";
        if (!jsonProj.has("SVNRepository")) return "";
        boolean hasJava = false;
        ... // is the project a Java project?
        if (!hasJava) return "";
        JSONObject svnRep = jsonProj.getJSONObject("SVNRepository");
        if (!svnRep.has("location")) return "";
        return svnRep.getString("location");
    }

    private double getChurnRateForProject(String url) {
        double rate = 0;
        SVNURL svnUrl;
        ... // connect to SVN and compute churn rate
        return rate;
    }
}

Full program
over 70 lines of code

Uses JSON and SVN libraries

Runs sequentially

Takes over 24 hrs

Takes almost 3 hrs - with data locally cached!
A better solution...

```
p: Project = input;
rates: output mean[string] of int;

exists (i: int; lowercase(p.programming_languages[i]) == "java")
  foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
    foreach (k: int; def(p.code_repositories[j].revisions[k]))
      rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```

Full program **6 lines of code!**

**Automatically parallelized!**

**No external libraries** needed!

Results in about **1 minute!**
A better solution...

```
p: Project = input;
rates: output mean[string] of int;

exists (i: int; lowercase(p.programming_languages[i]) == "java")
    foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
        foreach (k: int; def(p.code_repositories[j].revisions[k]))
            rates[p.id] << len(p.code_repositories[j].revisions[k].files);
```
The Boa language and data-intensive infrastructure

http://boa.cs.iastate.edu/
Research Questions

1. Can we abstract and simplify the software mining process to make it more accessible to non-experts?

2. Can software repository mining be done efficiently at a large scale?
Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Design goals

Easy to use

- Simple language
- No need to know details of
  - Software repository mining
  - Data parallelization
Design goals

Scalable and efficient

- Study *millions* of projects
- Results in minutes, not days
Reproducible research results

Robles, MSR'10

Studied 171 papers

Only 2 were "replication friendly"

Reproducible research results

Robles, MSR'10

Studied 171 papers

Only 2 were "replication friendly"
Boa architecture

**Boa Language**
- MapReduce
- Domain-specific Types/Functions

**Boa's Compiler**
- MapReduce
- Quantifiers
- User Functions
- Cached Data
- Runtime

**Boa's Data Infrastructure**
- SF.net
  - Replicator
  - Caching Translator
- Local Cache

1 Pike et al, Scientific Prog. Journal, Vol 13, No 4, 2005
Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Domain-specific types

http://boa.cs.iastate.edu/docs/dsl-types.php

p: Project = input;
rates: output mean[string] of int;

exists (i: int; lowercase(p.programming_languages[i]) == "java")
    foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
        foreach (k: int; def(p.code_repositories[j].revisions[k]))
            rates[p.id] << len(p.code_repositories[j].revisions[k].files);

Abstracts details of how to mine software repositories
Domain-specific types

http://boa.cs.iastate.edu/docs/dsl-types.php

Project

id : string
name : string
description : string
homepage_url : string
programming_languages : array of string
licenses : array of string
maintainers : array of Person
....
code_repositories : array of CodeRepository
## Domain-specific types

http://boa.cs.iastate.edu/docs/dsl-types.php

### CodeRepository

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>string</td>
</tr>
<tr>
<td>kind</td>
<td>RepositoryKind</td>
</tr>
<tr>
<td>revisions</td>
<td>array of Revision</td>
</tr>
</tbody>
</table>

### Revision

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int</td>
</tr>
<tr>
<td>committer</td>
<td>Person</td>
</tr>
<tr>
<td>commit_date</td>
<td>time</td>
</tr>
<tr>
<td>log</td>
<td>string</td>
</tr>
<tr>
<td>files</td>
<td>array of File</td>
</tr>
</tbody>
</table>

### File

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>kind</td>
<td>FileKind</td>
</tr>
<tr>
<td>change</td>
<td>ChangeKind</td>
</tr>
</tbody>
</table>
Domain-specific functions

http://boa.cs.iastate.edu/docs/dsl-functions.php

```haskell
hasfiletype := function (rev: Revision, ext: string) : bool {
    exists (i: int; matches(format(`\.%s$`, ext), rev.files[i].name))
        return true;
    return false;
}
```

Mines a revision to see if it contains any files of the type specified.
isfixingrevision := function (log: string) : bool {
    if (matches(`\s+fix(es|ing|ed)?\s+`, log)) return true;
    if (matches(`(bug|issue)(s)?\s+[#]?\s*\[0-9\]+`, log)) return true;
    if (matches(`(bug|issue)\s+id(s)?\s*\=\s*\[0-9\]+`, log)) return true;
    return false;
}

Mines a revision log to see if it fixed a bug.
User-defined functions

http://boa.cs.iastate.edu/docs/user-functions.php

id := function (a_1: t_1, ..., a_n: t_n) [:: ret] {
    ...
    # body
    [return ...;]
};

- Allows for complex algorithms and code re-use
- Users can provide their own mining algorithms
Quantifiers
http://boa.cs.iastate.edu/docs/quantifiers.php

p: Project = input;
rates: output mean[string] of int;

exists (i: int; lowercase(p.programming_languages[i]) == "java")
    foreach (j: int; p.code_repositories[j].kind == RepositoryKind.SVN)
        foreach (k: int; def(p.code_repositories[j].revisions[k]))
            rates[p.id] << len(p.code_repositories[j].revisions[k].files);

- foreach, exists, ifall
- Bounds are inferred from the conditional
Output and aggregation

http://boa.cs.iastate.edu/docs/aggregators.php

\[ p: \text{Project} = \text{input}; \]
\[ \text{rates: output mean[string] of int;} \]

\[ \text{exists } (i: \text{int}; \text{lowercase}(p.\text{programming_languages}[i]) == "java") \]
\[ \text{foreach } (j: \text{int}; p.\text{code_repositories}[j].\text{kind} == \text{RepositoryKind.SVN}) \]
\[ \text{foreach } (k: \text{int}; \text{def}(p.\text{code_repositories}[j].\text{revisions}[k])) \]
\[ \text{rates}[p.\text{id}] \ll \text{len}(p.\text{code_repositories}[j].\text{revisions}[k].\text{files}); \]

- Output can be indexed
- Output defined in terms of predefined data aggregators
  - sum, set, mean, maximum, minimum, etc
- Values sent to output aggregation variables
Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
Let's see it in action!

<<demo>>
Why are we waiting for results?

Program is analyzing...

- 699,332 projects
- 494,159 repositories
- 6,385,666 revisions
- 57,304,233 files
Let's check the results!

<<demo>>
Efficient execution
Scalability of input size

![Scalability graph showing the total time (seconds) for different input sizes and tasks. The graph compares Boa and Java for tasks 1 to 4.]
Design goals

- Easy to use
- Scalable and efficient
- Reproducible research results
# Controlled Experiment

- Published artifacts (on Boa website)
  - Boa source code
  - Dataset used (timestamp of data)
  - Results file

<table>
<thead>
<tr>
<th>Expert</th>
<th>Education</th>
<th>Intro</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Post-doc</td>
<td>6</td>
<td>B.1</td>
<td>B.6</td>
<td>B.9</td>
</tr>
<tr>
<td>Yes</td>
<td>PhD</td>
<td>5</td>
<td>A.1</td>
<td>B.6</td>
<td>B.7</td>
</tr>
<tr>
<td>No</td>
<td>PhD</td>
<td>4</td>
<td>B.6</td>
<td>B.10</td>
<td>B.9</td>
</tr>
<tr>
<td>No</td>
<td>PhD</td>
<td>4</td>
<td>A.2</td>
<td>B.6</td>
<td>D.5</td>
</tr>
<tr>
<td>No</td>
<td>MS</td>
<td>4</td>
<td>A.1</td>
<td>B.6</td>
<td>D.3</td>
</tr>
<tr>
<td>No</td>
<td>MS</td>
<td>3</td>
<td>B.6</td>
<td>C.1</td>
<td>D.4</td>
</tr>
<tr>
<td>No</td>
<td>MS</td>
<td>6</td>
<td>A.1</td>
<td>B.7</td>
<td>B.10</td>
</tr>
<tr>
<td>No</td>
<td>BS</td>
<td>2</td>
<td>A.2</td>
<td>D.1</td>
<td>D.3</td>
</tr>
</tbody>
</table>

Fig. 16. Study results. All times given in minutes.
Related Works

**Sourcerer** [Linstead et al. Data Mining Know. Disc.'09]
- SQL database on 18k projects

**Kenyon** [Bevan et al. ESEC/FSE'05]
- Centralized database of metadata and source code

**PROMISE** [Boetticher, Menzies, Ostrand 2007]
- Online data repository for SE datasets
- Boa provides raw, un-processed data

Boa provides better scalability
Related Works

Sawzall  [Pike et al. Sci.Prog.’05]
  ● Similar syntax to Boa
  ● Abstracts details of the MapReduce runtime

Pig Latin  [Olston et al. SIGMOD’08]
  ● Declarative syntax, similar to SQL

DryadLINQ  [Yu et al. OSDI’08]
  ● Syntax based on .Net's LINQ
  ● Compiles to Dryad framework, a DAG of processes

None provide direct support for mining software repositories
Ongoing work

Infrastructure improvements

Language abstractions

Other artifacts

cvs

git

bzc

hg

GitHub

Google Code

Launchpad

Other artifacts

Language abstractions

Infrastructure improvements

Ongoing work
Recent Work

● Support for mining source code
  ○ Down to expression level

● Currently for Java
  ○ Over 23k projects, with full history
  ○ Over 14 Billion AST nodes
Conclusions

- Domain-specific language and infrastructure for software repository mining
  - Easy to use
  - Efficient and scalable
  - Allows reproducing prior results

http://boa.cs.iastate.edu/request/