Tools for large-scale collection & analysis of source code repositories
Alexander Bezzubov

➔ committer & PMC @ apache zeppelin
➔ engineer @source{d}

source{d}

➔ startup in Madrid
➔ builds the open-source components that enable large-scale code analysis and machine learning on source code
MOTIVATION: WHY COLLECTING SOURCE CODE

➔ Academia: material for research in IR/ML/PL communities

➔ Industry: fuel for building data-driven products (i.e. for sourcing candidates for hiring)

VISION:

➔ OSS collection pipeline

➔ Use it to build public datasets industry and academia

➔ Use Git as “source of truth”, the most popular VCS

➔ A crawler (find URLs, git clone them), Distributed storage (FS, DB), Parallel processing framework

  custom, in Golang  standard, Apache HDFS + Postgres  custom, library for Apache Spark
Tech Stack
<table>
<thead>
<tr>
<th>tech stack</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>infrastructure</strong></td>
<td>CoreOS, K8s</td>
</tr>
<tr>
<td><strong>collection</strong></td>
<td>Rovers, Borders, go-git</td>
</tr>
<tr>
<td><strong>storage</strong></td>
<td>HDFS, śiva</td>
</tr>
<tr>
<td><strong>processing</strong></td>
<td>Apache Spark, source{d} Engine</td>
</tr>
<tr>
<td><strong>analysis</strong></td>
<td>Bblfsh, Enry</td>
</tr>
<tr>
<td>tech stack</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>CoreOS</td>
</tr>
<tr>
<td></td>
<td>K8s</td>
</tr>
</tbody>
</table>
[infrastructure]

- Dedicated cluster (cloud becomes prohibitively expensive for storing ~100sTb)
- CoreOS provisioned on bare-menta \w Terraform
- Booting and OS configuration Matchbox and Ignition
- K8s deployed on top of that

More details at talk at CfgMgmtCamp
http://cfgmgmtcamp.eu/schedule/terraform/CoreOS.html
<table>
<thead>
<tr>
<th>tech stack</th>
<th>infrastructure</th>
<th>collection</th>
<th>storage</th>
<th>processing</th>
<th>analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CoreOS</td>
<td>Rovers</td>
<td>HDFS</td>
<td>Apache Spark</td>
<td>Bblfsh</td>
</tr>
<tr>
<td></td>
<td>K8s</td>
<td>Borders</td>
<td>šiva</td>
<td>source{d} Engine</td>
<td>Enry</td>
</tr>
</tbody>
</table>
Rovers: search for Git repository URLs
Borges: fetching repository \w “git pull”
Git storage format & protocol implementation
Optimize for on-disk size: forks that share history, saved together

go-git to talk Git Last year had a talk at FOSDEM
https://archive.fosdem.org/2017/schedule/event/go_git/
motivation

GIT LIBRARY FOR GO

• need to clone and analyze tens of millions of repositories with our core language Go
• be able to do so in memory, and by using custom filesystem implementations
• easy to use and stable API for the Go community
• used in production by companies, e.g.: keybase.io

features

PURE GO SOURCE CODE

• the most complete git library for any language after libgit2 and jgit
• highly extensible by design
• idiomatic API for plumbing and porcelain commands
• 2+ years of continuous development
• used by a significant number of open source projects

example

GO-GIT IN ACTION

example mimicking `git clone` using go-git:

```go
// Clone the repo to the given directory
url := "https://github.com/src-d/go-git",
_, err := git.PlainClone(
    "/tmp/foo", false,
    &git.CloneOptions{
        URL: url,
        Progress: os.Stdout,
    },
)
CheckIfError(err)
```

output:

```
Counting objects: 4924, done.
Compressing objects: 100% (1333/1333), done.
Total 4924 (delta 530), reused 6 (delta 6),
pack-reused 3533
```

usage

TRY IT YOURSELF

```bash
$ go get -u gopkg.in/src-d/go-git.v4/...
```

• list of more go-git usage examples

resources

YOUR NEXT STEPS

• https://github.com/src-d/go-git
• go-git presentation at FOSDEM 2017
• go-git presentation at Git Merge 2017
• compatibility table of git vs. go-git
• comparing git trees in go
**motivation**

CODE COLLECTION AT SCALE

- collection and storage of repositories at large scale
- automated process
- optimal usage of storage
- optimal to keep repositories up-to-date with the origin

---

**architecture**

SEEK, FETCH, STORE

- distributed system similar to a search engine
- **src-d/rovers** retrieves URLs from git hosting providers via API, plus self-hosted git repositories
- **src-d/borges** producer reads URL list, schedules fetching
- borges consumer fetches and pushes repo to storage
- borges packer also available as a standalone command, transforming repository urls into siva files
- stores using **src-d/śiva** repository storage file format
- optimized for storage and keeping repos up-to-date

---

**key concept**

- **rooted repositories** are standard git repositories that store all objects from all repositories that share a common history, identified by same initial commit:

- a rooted repository is saved in a single śiva file
- updates stored in concatenated siva files: no need to rewriting the whole repository file
- distributed-file-system backed, supports GCS & HDFS

---

**usage**

SETUP & RUN

- set up and run rovers
- set up borges
- run borges producer
- run borges consumer

---

**resources**

YOUR NEXT STEPS

- [https://github.com/src-d/rovers](https://github.com/src-d/rovers)
- [https://github.com/src-d/borges](https://github.com/src-d/borges)
- [https://github.com/src-d/go-siva](https://github.com/src-d/go-siva)
- śiva: Why We Created Yet Another Archive Format
<table>
<thead>
<tr>
<th>tech stack</th>
<th>CoreOS</th>
<th>K8s</th>
</tr>
</thead>
<tbody>
<tr>
<td>infrastructure</td>
<td>Rovers</td>
<td>Borders</td>
</tr>
<tr>
<td>collection</td>
<td>go-git</td>
<td></td>
</tr>
<tr>
<td>storage</td>
<td>HDFS</td>
<td>śiva</td>
</tr>
<tr>
<td>processing</td>
<td>Apache Spark</td>
<td>source{d} Engine</td>
</tr>
<tr>
<td>analysis</td>
<td>Bblfsh</td>
<td>Enry</td>
</tr>
</tbody>
</table>
→ Metadata: PostgreSQL

→ Built small type-safe ORM for Go<->Postgres

   https://github.com/src-d/go-kallax

→ Data: Apache Hadoop HDFS

→ Custom (seekable, appendable) archive format: Siva 1 RootedRepository <-> 1 Siva file
**motivation**

SMART REPO STORAGE

- store a git repository in a single file
- updates possible without rewriting the whole file
- friendly to distributed file systems
- seekable to allow random access to any file position

**architecture**

SIVA FILE BLOCK SCHEMA

**CHARACTERISTICS**

- src-d/go-siva is an archiving format similar to tar or zip
- allows constant-time random file access
- allows seekable read access to the contained files
- allows file concatenation given the block-based design
- command-line tool + implementations in Go and Java

**usage**

**APPENDING FILES**

```
# pack into siva file
$ siva pack example.siva qux

# append into siva file
$ siva pack --append example.siva bar

# list siva file contents
$ siva list example.siva
Sep 20 13:04  4 B qux -rw-r--r--
Sep 20 13:07  4 B bar -rw-r--r--
```
<table>
<thead>
<tr>
<th>tech stack</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>infrastructure</strong></td>
<td>Core OS</td>
<td>K8s</td>
</tr>
<tr>
<td><strong>collection</strong></td>
<td>Rovers</td>
<td>Borders</td>
</tr>
<tr>
<td><strong>storage</strong></td>
<td>HDFS</td>
<td>śiva</td>
</tr>
<tr>
<td><strong>processing</strong></td>
<td>Apache Spark</td>
<td>source{d} Engine</td>
</tr>
<tr>
<td><strong>analysis</strong></td>
<td>Bblfsh</td>
<td>Enry</td>
</tr>
</tbody>
</table>
Apache Spark

- For batch processing, SparkSQL

Engine

- Library, \w custom DataSource implementation GitDataSource
- Read repositories from Siva archives in HDFS, exposes though DataFrame
- API for accessing refs/commits/files/blobs
- Talks to external services though gRPC for parsing/lexing, and other analysis
motivation

UNIFIED SCALABLE PIPELINE

• easy-to-use pipeline for git repository analysis
• integrated with standard tools for large scale data analysis
• avoid custom code in operations across millions of repos

architecture

PREPARATION

• listing and retrieval of git repositories
• Apache Spark datasource on top of git repositories
• iterators over any git object, references
• code exploration and querying using XPath expressions
• language identification and source code parsing
• feature extraction for machine learning at scale

• extends Apache SparkSQL
• git repositories stored as siva files or standard repositories in HDFS
• metadata caching for faster lookups over all the dataset.
• fetches repositories in batches and on demand
• available APIs for Spark and PySpark
• can run either locally or in a distributed cluster

usage sample

```python
EngineAPI(spark, 'siva', '/path/to/siva-files')
.repos
.refernces
.files
.classify_languages()
.extract_uasts()
.query_uast('//[@roleImport and @roleDeclaration]', 'imports')
.filter('lang = 'java')
.select('imports', 'path', 'repository_id')
.write.parquet('hdfs://...')
```

resources

YOUR NEXT STEPS

• https://github.com/src-d/engine
• Early example jupyter notebook: https://github.com/src-d/spark-api/blob/master/examples/notebooks/Example.ipynb
<table>
<thead>
<tr>
<th>tech stack</th>
<th>CoreOS</th>
<th>K8s</th>
</tr>
</thead>
<tbody>
<tr>
<td>infrastructure</td>
<td>Rovers</td>
<td>Borders</td>
</tr>
<tr>
<td>collection</td>
<td>go-git</td>
<td></td>
</tr>
<tr>
<td>storage</td>
<td>HDFS</td>
<td>Šiva</td>
</tr>
<tr>
<td>processing</td>
<td>Apache Spark</td>
<td>source{d} Engine</td>
</tr>
<tr>
<td>analysis</td>
<td>Bblfsh</td>
<td>Enry</td>
</tr>
</tbody>
</table>
Enry

→ Programming language identification

→ Re-write of github/linguist in Golang, ~370 langs

Project Babelfish

→ Distributed parser infrastructure for source code analysis

→ Unified interface though gRPC to native parsers in containers: src -> uAST

Talk in Source Code Analysis devRoom
Room: UD2.119, Sunday, 12:40
https://fosdem.org/2018/schedule/event/code_babelfish_a_universal_code_parser_for_source_code_analysis/
**motivation**

**LANG DETECTION AT SCALE**

- need to detect programming languages of every file in a git repository
- initially used github/linguist, but needed more performance for large scale applications
- keep compatibility with the original linguist project

**architecture**

**COMPATIBLE AND FLEXIBLE**

- linguist as source of information on language detection
- ignores binary and vendored files
- command line tool mimics the original linguist one
- can be used in Go (native library) or Java (shared library)

**benchmarks**

**GO FASTER**

- src-d/enry is at least 4x faster than linguist
- 5x (larger repos) to 20x faster (smaller repos)

![Histogram of processing time](image)

**usage**

usable in Go as a native library, in Java as shared library and as a CLI tool.

**resources**

**YOUR NEXT STEPS**

- https://github.com/src-d/enry
- enry: detecting languages
- benchmark methodology and results
**motivation**

**UNIVERSAL CODE ANALYSIS**

- was born as a solution for massive code analysis
- parsing single files in any programming language
- analyze all source code from all repositories in the world
- analyze many languages using a shared structure/format

**use cases**

**POWERFUL OPPORTUNITIES**

- AST-based diff'ing. Understanding changes made to code with finer-grained granularity.
- extract features for Machine Learning on Source Code.
- statistics of language features
- detecting similar coding patterns across languages

**architecture**

**CONTAINER-BASED**

- language drivers as the main building blocks
- parsing service via one driver per language
- language drivers can be written in any language and are packaged as standard Docker containers
- containers are executed by the babelfish server in a specific runtime built on-top of libcontainer.

**resources**

**YOUR NEXT STEPS**

- https://github.com/bblfsh
- Babelfish documentation
- announcing Babelfish
- Babelfish presentation
- join the Babelfish community

**architecture**

**UNIVERSAL AST**

- UAST is a universal (normalized and annotated) form of Abstract Syntax Tree (AST)
- language-independent annotations (roles) such as Expression, Statement, Operator, Arithmetic, etc.
- can be easily ported to many languages using gogo/protobuf

**usage**

**TRY BABELFISH ONLINE**

- or run babelfish server & dashboard locally:

  ```bash
  $ docker run --privileged -d -p 9432:9432 --name bblfsh bblfsh/server
  $ docker run -p 8080:80 --link bblfsh bblfsh/dashboard --bblfsh-addr bblfsh:9432
  ```
### tech stack

<table>
<thead>
<tr>
<th>Infrastructure</th>
<th>CoreOS</th>
<th>K8s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>Rovers</td>
<td>Borders</td>
</tr>
<tr>
<td></td>
<td>go-git</td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>HDFS</td>
<td>śiva</td>
</tr>
<tr>
<td>Processing</td>
<td>Apache Spark</td>
<td>source{d} Engine</td>
</tr>
<tr>
<td>Analysis</td>
<td>Bblfsh</td>
<td>Enry</td>
</tr>
</tbody>
</table>
Further directions
further directions

INFRASTRUCTURE

➔ Persistent storage in k8s on bare-metal cluster

COLLECTION

➔ Explore SEDA architecture, to dynamically saturate throughput

STORAGE

➔ Better splittable Git object storage format (\w delta-encoding, etc)

PROCESSING

➔ Distributed Indexes to speed up common Apache Spark queries

ANALYSIS

➔ AST-diff, cross-language abstractions on top of ASTs
thank you.

source{d}