Boidae: Your Personal Mining Platform

Brian Sigurdson
Bowling Green State University
Bowling Green, Ohio
bsigurd@bgsu.edu

Robert Dyer
Bowling Green State University
Bowling Green, Ohio
rdyer@bgsu.edu

ABSTRACT

Boa provides users with the ability to easily mine ultra-large-scale open source software repositories. The trade-off is users must use the provided infrastructure, language, runtime, and datasets. For users who can't work with these limitations, in this work we provide Boidae: a family of Boa installations controlled and customized by users. Boidae uses automation tools such as Ansible to facilitate deployment of Boa installations onto CloudLab. Here we demo how even novices with little to no knowledge of cloud deployments, Hadoop, SQL, Drupal, etc are able to instantiate their own distributed cluster running Boa in about 20 minutes.
YouTube video: https://boa.page.link/BoidaeFSE18

CCS CONCEPTS

• **Software and its engineering** → *Concurrent programming structures*;

KEYWORDS

Boa, mining software repositories, scalable, open source

ACM Reference Format:
Brian Sigurdson and Robert Dyer. 2018. Boidae: Your Personal Mining Platform. In *Proceedings of The 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Boa [2, 3, 6] is a domain-specific language and infrastructure designed to ease mining software repository (MSR) research. Boa has been shown to reduce the effort necessary for researchers to mine ultra-large-scale repositories such as GitHub and SourceForge. Boa uses Hadoop for storage and computation, and therefore is able to scale to accommodate very large datasets. It provides users with a web-based interface and allows users to easily replicate Boa experiments provided by other researchers.

For researchers wishing to analyze the datasets provided by Boa and utilize the default language and runtime provided by the infrastructure, Boa works great. However, if someone wishes to analyze a different dataset (perhaps private/industrial data) or wishes to modify the language or runtime (to add new features or analysis capabilities) this is currently not feasible with Boa.

ESEC/FSE 2018, 4–9 November, 2018, Lake Buena Vista, Florida, United States

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this work, we present Boidae. Boidae provides users the ability to instantiate the entire Boa infrastructure on their own cloud infrastructure. For the initial prototype, we chose CloudLab [1, 7] as the supported cloud infrastructure. CloudLab is an NSF funded meta-cloud facility, that provides researchers with the ability to provision bare-metal servers and configure them at will. This infrastructure is currently free to all researchers who request an account.

Boidae uses the configuration management and automation tool Ansible to automatically install and configure all required software on the nodes of the cluster. Once the cluster is fully booted and configured, users have a working installation of Boa with a small sample dataset installed. They can then proceed to configure it as they wish.

Software Engineering (SE) Challenges Addressed Boidae provides SE and MSR researchers with an easy to install and automatically configured infrastructure that is scalable to their needs. Users can create datasets based on their own data and use the Boa infrastructure to analyze it.

Envisioned Users While Boa aims to provide an easy to use, single place for users wishing to mine open source repositories, Boidae's intended users are more advanced software miners who need customization not currently feasible with Boa.

Methodology Implied For Users Users create accounts on CloudLab, then instantiate the Boidae profile. The infrastructure starts a cluster and automatically configures all software, providing a working Boa instance that can then be further customized.

Planned Study To evaluate the usefulness of Boidae, in the future we are planning a user study to identify how easy it is to use and what portions of the deployment users needed to manually modify.

Source Availability All of the code for this project is open-source and located at: https://github.com/boalang/ansible

2 RELATED WORK

Boa [2, 3, 6] is an infrastructure and language for easily mining ultra-large-scale software repositories such as GitHub. Boidae is an automated installation of the Boa infrastructure.

GHTorrent [4, 5] is a website and dataset, updated every several months, containing the event stream from GitHub. Similar to Boa, users are able to use a shared infrastructure if they wish, but can not easily create their own scalable infrastructure based off it without having substantial expertise.

Cloud infrastructures like Google's BigQuery can also provide the ability to easily query large amounts of data. GitHub even has large event stream datasets hosted on BigQuery.

3 BOIDAE ARCHITECTURE

To instantiate Boidae, the following steps are taken:

(0) provision and boot cluster nodes (handled by CloudLab);
(1) install Ansible and create the user `ansible`;
(2) install and configure Hadoop;
(3) retrieve Boa-specific files; and
(4) install Drupal and enable Boa's Drupal module.

3.1 Ansible Setup

The first step to deploy Boidae on CloudLab is for the user to start an experiment and select the `boidae` profile, as shown in Figure 1.

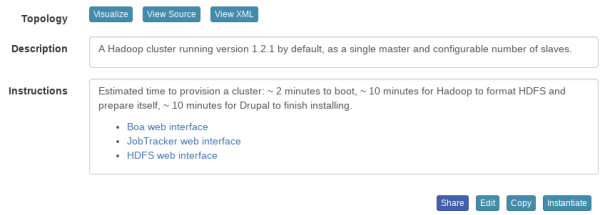

Figure 1: CloudLab `boidae` profile information

In this figure, the user has selected the profile and is presented with summary information and further options. Selecting the instantiate button will bring the user to an input page, where they simply enter the number of slave nodes for their cluster (see Figure 2).

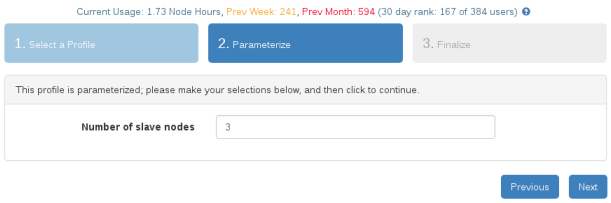

Figure 2: User specified number of managed nodes

After indicating the number of slave nodes in the cluster the user is presented with Figure 3, where they can choose from several locations from which to deploy their cluster.

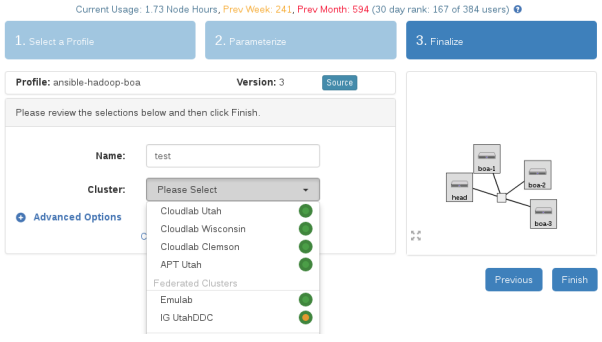

Figure 3: User selects location to deploy cluster

CloudLab profiles allow programmatic customization of the boot sequence. In our profile, ClouldLab is instructed to download the following tar file from the project's GitHub repository into the server's `/tmp` directory: http://github.com/boalang/ansible/archive/master.tar.gz. This compressed file is then automatically extracted by CloudLab and a script executed as root.

A different execution path is followed depending on whether the node is the master node or a slave node. Although, all nodes experience the following modifications:

- installation of Python 2.7;
- creation of a user named `ansible`;
- user `ansible` is given passwordless sudoer privileges; and
- creation of ssh keys for the user `ansible`.

At a high level, the differences between the setup of the master and salve nodes comes down to the master node having the Ansible software installed on it. This is to ease the management of the slave nodes during the installation and configuration process of Hadoop, Boa, and Drupal. The master node also collects the public ssh keys from the slave nodes to facilitate communication. At this point, the setup of Ansible and an `ansible` user is complete.

3.2 Hadoop Setup

The setup and installation of Hadoop goes through the following stages:

(1) Create Ansible Hosts / Inventory file
(2) Retrieve Hadoop files
(3) Deploy Hadoop compressed files
(4) Setup Secondary Name Node configuration
(5) Format Hadoop
(6) Install Boa related Hadoop files
(7) Start Hadoop

3.2.1 Hosts File. The first step is to create a host file for Ansible. A host file, also known as an inventory file, provides Ansible with a mapping of group names to server names or server IP address. This mapping allows the user to act on groups of servers by executing commands against a single group name. Additional information, such as a specified port, can be included also. Servers can be in more than one group, such as a web server that is also a database server. In our case, each server is either the master or a slave node. The host file is created on the fly using bash, based on a predetermined naming scheme, and the number of managed nodes indicated by the user.

3.2.2 Retrieve Hadoop Files. Although Ansible can be run from the command line, it is more common to define a set of tasks in a YAML file and let Ansible act on that file. Ansible refers to these YAML files as playbooks. The simple playbooks the we've used are generally divided into two parts: Playbook configuration settings and tasks to complete.

Figure 4 illustrates the playbook configuration setting for a playbook called `compressed-file-setup.yml`, which is the playbook that runs and retrieves the Hadoop compressed files. One of the benefits of using YAML files is their general readability. Looking at Figure 4 you can see the various configuration settings for this playbook.

```
############################################################
# start playbook
############################################################

- name: Compressed file setup for Hadoop {{ hadoop_version }}
  hosts: name_node
  remote_user: ansible
  become_user: root
  become: true
  become_method: sudo
  connection: ssh
  gather_facts: no

  vars_files:
    - ../local_variable_files/hadoop-vars.yml
```

Figure 4: Configuration of the compressed-file-setup.yml playbook

```
#########################################################################
# start tasks
#########################################################################

  tasks:
  - name: create {{ hadoop_compressed }}, if it does not exist
    file:
      path: "{{ hadoop_compressed }}"
      state: directory
      mode: 0755
      recurse: yes


  - name: Test if "{{ hadoop_compressed }}/{{ hadoop_file }}" already exists.
    stat:
      path: "{{ hadoop_compressed }}/{{ hadoop_file }}"
    register: hd_file
```

Figure 5: Example tasks from compressed-file-setup.yml playbook

A similar key/value pair structure is employed for the playbook tasks, a sample of which can be seen in Figure 5. Tasks often begin with the "name:" item to provide human readable output to the user while running the playbook. Figure 5 illustrates the first two tasks in `compressed-file-setup.yml`. The top task executes the Ansible module called `file`, which creates the compressed file directory structure, if it does not already exist. The second task tests for the existence of the desired Hadoop compressed file, and stores the result into the variable `hd_file`. This variable can be inspected in subsequent tasks to determine different execution paths based on the contents of the variable.

While Ansible is used throughout the rest of the Boidae deployment, subsequent playbooks will not be discussed in as much detail as we have done here. This was merely to provide a bit of background information to users unfamiliar with Ansible.

3.2.3 Deploy Hadoop Compressed Files. Once the Hadoop compressed files have been retrieved and checksums tested, the next step is to deploy the necessary files to the nodes in the cluster. The installation playbook will create the user necessary to manage Hadoop in each node. In our case the user is simply named `hadoop`. The playbook will execute the following steps on each cluster nodes:

(1) create the user hadoop to manage Hadoop;
(2) copy and extract Hadoop's compressed file on each node;
(3) set appropriate file permissions;
(4) set needed environment variables;
(5) create the necessary directory structure for Hadoop's HDFS file system; and
(6) create the necessary template files, such as core-site.xml and hdfs-site.xml.

3.2.4 Setup Secondary Name Node. The next step is to setup the secondary namenode, which will regularly retrieve and merge the master node's image file with the master node's edit file. This is to prevent the edit file from growing too large and causing the next startup of the cluster to be excessively long. We designated the master to be its own secondary namenode. This is fine for short-lived deployments such as this, but production systems would normally have a different node designated for such purposes.

3.2.5 Format Hadoop HDFS. At this point we can format Hadoop's HDFS file system in anticipation of its first use. Formatting creates the directory structures and initial data structures for the file system. The formatting process is only conducted on the master node, as the master node manages the cluster's meta-data. The size of the file system is not specified, because that is determined by the number of nodes in the cluster, which can change dynamically.

3.2.6 Install Boa Related Hadoop Files. We now run another playbook to retrieve various Boa dependencies for Hadoop. Items such as `mysql-connector-java.jar` and `protobuf-java-2.5.0.jar` are retrieved and installed into the appropriate directory. This also installs the Snappy compression library and enables it for Hadoop.

3.2.7 Start Hadoop. At this point the `start-stop.yml` playbook is run to start the appropriate Hadooop daemons (namenode, jobtracker, secondarynamenode, datanode, tasktracker).

3.3 Boa Setup

The `setup-boa.yml` playbook runs to retrieve and install the remaining Boa items. The following items are retrieved and installed in the appropriate location:

- BoaCompilePoller.java - polls for new Boa web jobs
- run-poller.sh - starts the poller
- boa-compile.sh - script to compile Boa programs
- boa-run.sh - script to run Boa programs
- boa-compiler.jar - the Boa compiler
- boa-runtime.jar - the Boa runtime
- projects.seq - a small sample dataset

Cron jobs are then created to ensure the BoaCompilePoller is always running.

3.4 Drupal Setup

Now that Ansible, Hadoop, and Boa have been installed and configured, we proceed to the last step of the process and install Drupal and Boa's Drupal module.

The `install-drupal.yml` playbook installs Apache, MySQL, and PHP using modules provided by Jeff Geerling[1]. These and other modules can be found in Ansible's public repository called Galaxy https://galaxy.ansible.com/.

[1] https://galaxy.ansible.com/geerlingguy/apache/

The playbook also installs Drush 7[2], sets the appropriate configuration files, creates Boa's webroot, then installs and enables the Boa Drupal module.

Once the installation is complete, the user can access the login page. The default username is `boa` and default password is `rocks`. Once logged in, the user can access any of the tabs, but the tab of most interest is likely to be the Run Examples tab.

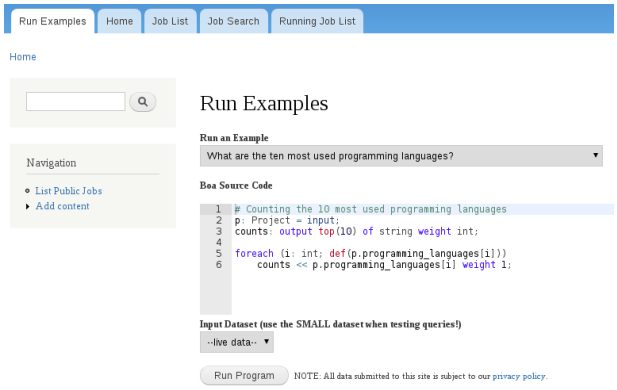

Figure 6: User selects example to run

Boa's Drupal interface provides many example queries. In Figure 6, the user has selected to run the query "What are the ten most used programming languages?" on the sample dataset included with the installation.

The Boa source code for the query is displayed in the middle of Figure 6. Once the user has selected the Run Program button, the code will be compiled and executed against the sample data and the output can be viewed.

4 FUTURE WORK

Boidae is in the process of being extended to Amazon Web Services (AWS) [8]. Although CloudLab has many benefits, such as being free, it has an inherent geographic drawback. All of the current facilities are located in the United States. Figure 7 displays the location of current Boa users.

Figure 7: Boa users' global presence

It is easy to see that the Boa user base is truly global in nature. Given the large size of even a modest repository dataset, mounting such a dataset in North America when the user is located internationally adds significant latency to interacting with CloudLab, and diminishes the user experience. There are also issues of data privacy that could make it difficult, or impossible, for some international users to interact with CloudLab.

[2] www.drush.org

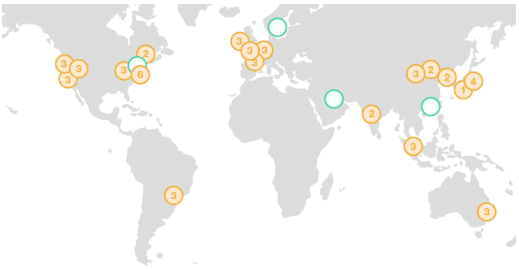

Figure 8: AWS global regions

AWS regions are shown in Figure 8. We believe AWS's global access points may provide a better option for many users. Therefore, the next step will be to expand Boidae's reach through AWS.

5 CONCLUSION

In this paper, we discussed many of the details to our approach of automating the deployment of a multi-node Boa cluster, which we call Boidae. The installation process is fully automated and allows users to instantiate the Boa infrastructure at will. This allows them to fully customize any portion of the infrastructure to their needs. In the future we plan to extend the approach, using the same scripts, to Amazon Web Services to provide more options to users.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under grants 1512947 and 1518776.

REFERENCES

[1] CloudLab. 2018. The CloudLab website. http://www.cloudlab.us/. (2018).
[2] Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N. Nguyen. 2013. Boa: A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories. In *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*. 422–431.
[3] Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N. Nguyen. 2015. Boa: Ultra-Large-Scale Software Repository and Source-Code Mining. *ACM Trans. Softw. Eng. Methodol.* 25, 1, Article 7 (2015), 7:1–7:34 pages.
[4] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*. IEEE Press, 233–236.
[5] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's Data from a Firehose. In *MSR '12: Proceedings of the 9th Working Conference on Mining Software Repositories*. IEEE, 12–21.
[6] Hridesh Rajan, Tien N. Nguyen, Robert Dyer, and Hoan Anh Nguyen. 2015. Boa website. http://boa.cs.iastate.edu/. (2015).
[7] Robert Ricci, Eric Eide, and The CloudLab Team. 2014. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *USENIX ;login:* 39, 6 (Dec. 2014). https://www.usenix.org/publications/login/dec14/ricci
[8] Amazon Web Services. 2018. AWS Global Infrastructure. (2018). Retrieved June 5, 2018 from https://aws.amazon.com/about-aws/global-infrastructure