Artifact Evaluation README

Blockaid: Data Access Policy Enforcement for Web Applications (OSDI '22)

(The submission title was *Embargo: Data Access Policy Enforcement for Web Applications*. The submission used the system name "Embargo" for blinding, and the system is referred to as Embargo in the comparison plots.)

Getting Started Instructions

The artifact consists of a Docker image that:

- Launches Amazon EC2 instances on your behalf to run the experiments.
- Generates Figure 5, Table 2, and Figure 8 from the paper.
- Produces a PDF that compares the generated figures with those from the paper.

Amazon EC2

The Blockaid experiments run on Amazon EC2 `c4.8xlarge` instances. By default, the artifact launches six such instances (totaling $36 \times 6 = 216$ vCPUs) to run experiments in parallel, although it provides an option to launch fewer instances at a time (see the Test Run section).

To be able to launch EC2 instances, the artifact requires AWS credentials with appropriate EC2 permissions (e.g., `AmazonEC2FullAccess` suffices). Please note down your AWS Access Key and Secret Key for later use.

By default, the artifact will launch EC2 instances in the **`us-east-2`** (Ohio) region from our public image `ami-01457e9b6b7cdee4e`, which contains Blockaid and the web applications under evaluation. If you would like to launch the experiments in another region, please copy the image to the desired region and note down its AMI ID. Alternatively, import this VM image (6.5 GB) as an AMI in your desired region: https://blockaid-ae.s3.us-east-2.amazonaws.com/export-ami-0d21f4fd779d80014.vmdk.

> Note down the following information before proceeding:
> - Your AWS access key.
> - Your AWS secret key.
> - AWS region (if not the default).
> - EC2 image AMI ID (if not the default).

Test Run

On a machine with Docker installed, pull the latest version of the Docker image:[1]

```
docker pull blockaid/ae:buildx-latest
```

Create a directory to store the output of a **test run**:

[1] We tested the Docker image on a Macbook Pro M1 and on a EC2 instance with the Ubuntu 20.04 (amd64) AMI.

```
mkdir /home/ubuntu/blockaid_test
```

Launch the Docker image with the directory mounted:

```
docker run -it --rm --name blockaid_test --mount \
  type=bind,source=/home/ubuntu/blockaid_test,target=/data \
  blockaid/ae:buildx-latest
```

You should now see a command prompt:

```
root@9bc6f02e8a8c:/app#
```

Let's start the experiment script in "test mode", which measures a small number of iterations. By default, the script launches six `c4.8xlarge` instances in parallel:

```
root@9bc6f02e8a8c:/app# ./run_all.sh test
```

You can adjust the degree of instance parallelism using the `PARALLEL` environment variable. For example, to run at most one EC2 instance at a time:

```
root@9bc6f02e8a8c:/app# PARALLEL=1 ./run_all.sh test
```

You will be prompted for your AWS credentials and other information:

```
AWS credentials not found...
AWS Access Key ID: xxxx ↵
AWS Secret Access Key: xxxx ↵
AWS region [default: us-east-2]: ↵
EC2 AMI ID [default: ami-01457e9b6b7cdee4e]: ↵
```

> If you need to later modify your AWS credentials, you can do so by manually editing the file `.credentials.sh` using the `nano` text editor, or by deleting `.credentials.sh` and re-running `run_all.sh`.

The script will now launch the test experiment on EC2, gather the results, delete the AWS resources, and produce a report file in the output directory named **`all_plots.pdf`**. This process should finish within **25 minutes** under the default configuration (i.e., full parallelism), and within 2 hours under sequential execution (i.e., `PARALLEL=1`). You may view the PDF the host machine (i.e., outside the container).

> If the script fails or is killed mid-way, you can manually clean up the AWS resources created for the experiment by calling `./cleanup.sh` within the container.

Inspect the report for comparisons between the figures generated from the experiment and figures from the submission. Because the test run has a shorter warmup phase, we expect the test report to indicate *higher latency across the board* than reported in the paper.

Exit the Docker container by pressing Ctrl-D.

Detailed Instructions

The artifact supports three modes, each taking a different amount of time:

Mode	Number of rounds					Expected duration	
	Original	Modified	Cached	Cold cache	No cache	Default (fully parallel)	PARALLEL=1
test	3	3	3	3	3	25 min	2 hr
small	300	300	300	10	10	1 hr 40 min	8 hr
full	3000	3000	3000	100	100	15 hr	

When starting an experiment, you can pass the mode name as a command line argument to `run_all.sh`, like we did with the `test` mode just now. The `full` mode corresponds to the experiment setup reported in the paper submission, and the `small` mode allows an experiment that is less accurate but faster than the full version.

You can choose to run either the `small` or the `full` experiment. Here is an example for running the `small` experiment (again, you can limit instance parallelism by passing the PARALLEL environment variable to `run_all.sh`):

```
mkdir /home/ubuntu/blockaid_small

docker run -it --rm --name blockaid_test --mount \
  type=bind,source=/home/ubuntu/blockaid_small,target=/data \
  blockaid/ae:buildx-latest

root@9bc6f02e8a8c:/app# ./run_all.sh small

AWS credentials not found...
AWS Access Key ID: xxxx ↵
AWS Secret Access Key: xxxx ↵
AWS region [default: us-east-2]: ↵
EC2 AMI ID [default: ami-01457e9b6b7cdee4e]: ↵

...
```

Once again, inspect the report at **`all_plots.pdf`** in the output directory. The smaller the run, the higher the measured latencies are expected to be due to fewer warmup iterations. However, for the `small` experiment (and, of course, the `full` experiment) we do expect the relative latencies to match what is reported in the submission – e.g., the relative positions of points in Figure 5. (The same cannot be said of the `test` experiment, which is too short to yield stable measurements.)