AWSOMEPY: A DATASET AND CHARACTERIZATION OF SERVERLESS APPLICATIONS

RAFFA, GIUSEPPE, JORGE BLASCO ALIS, DAN O'KEEFFE, AND SANTANU KUMAR DASH

DEC 05 2023 TCSS 562 Presentation

Review by: SHERRY LIU

KEWEI LIU

OUTLINE

Introduction of the paper

Related work

Methodology

Contributions

Evaluation

Critique

Q&A

INTRODUCTION

- Server management eliminated
- Stateless, Event-Driven Model
- Costs align with usage
- Focus shifts to product development, not infrastructure
- Dataset - AWSomePy

WHY CONCERN ABOUT THESE CHALLENGES?

- Performance optimization needed
- Event traceability is tough
- Security concerns are paramount

RELATED WORK

- Serverless Framework
- Wonderless: A dataset of 1,877 real-world Serverless applications.
- Limitations of Wonderless: Lacked focus on Python-based AWS applications.
- Why AWSomePy?:
  - Addresses gaps in Wonderless for Python-based AWS serverless applications.

[1] Nafise Eskandani and Guido Salvaneschi. 2021. The Wonderless Dataset for Serverless Computing. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). 565–569. https://doi.org/10.1109/MSR52588.2021. 00075

METHODOLOGY

Step 1. Configuration Files Gathering

Step 2. Configuration Files Filtering

Step 3. Repositories URLs Identification

Step 4. Repositories Filtering by Language & Cloning

Step 5. Repositories Filtering by Invalid Configuration

Step 6. Repositories Filtering by Immature Projects

Step 7. Repositories Filtering by Metadata

Step 8. Repositories Fork Analysis

Step 9. Metadata Gathering

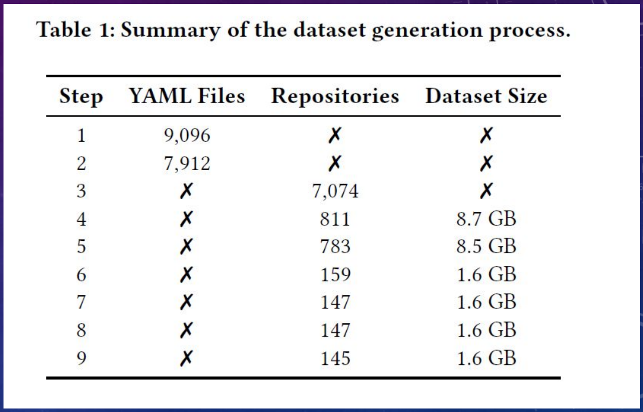


Table 1: Summary of the dataset generation process.

Step	YAML Files	Repositories	Dataset Size
1	9,096	✗	✗
2	7,912	✗	✗
3	✗	7,074	✗
4	✗	811	8.7 GB
5	✗	783	8.5 GB
6	✗	159	1.6 GB
7	✗	147	1.6 GB
8	✗	147	1.6 GB
9	✗	145	1.6 GB

Key Contribution

- New dataset - AWSomePy
- Characterization of the dataset
- Analysis of Serverless Applications
- Service and API Utilization Metrics
- Evaluation and Insights

Experimental Evaluation

Configuration & Architectural Analysis
Plugin analysis

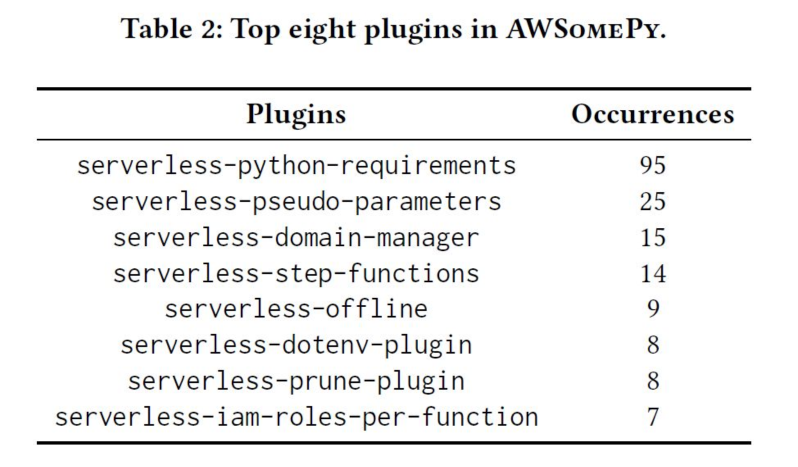


Table 2: Top eight plugins in AWSomePy.

Plugins	Occurrences
`serverless-python-requirements`	95
`serverless-pseudo-parameters`	25
`serverless-domain-manager`	15
`serverless-step-functions`	14
`serverless-offline`	9
`serverless-dotenv-plugin`	8
`serverless-prune-plugin`	8
`serverless-iam-roles-per-function`	7

Experimental Evaluation

Configuration & Architectural Analysis
Complexity analysis

Average: 4, 468
Min: 26
Max: 132, 658

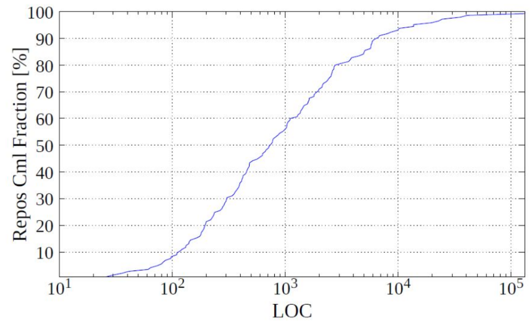


Figure 2: Cumulative distribution of the lines of code in AWSomePy (cumulative fraction of repositories on the y axis).

Experimental Evaluation

Configuration & Architectural Analysis
Complexity analysis

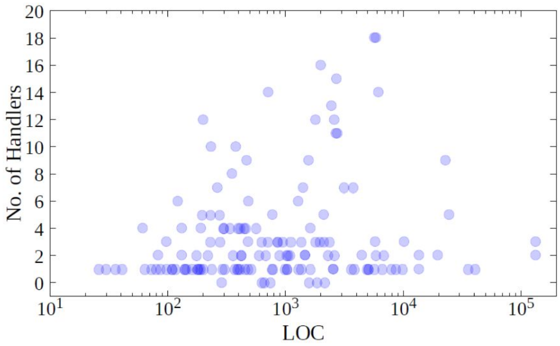


Figure 3: Number of handlers vs lines of code in AWSomePy.

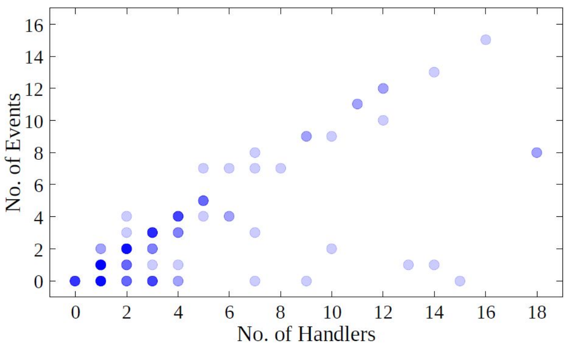


Figure 4: Number of events vs handlers in AWSomePy.

Experimental Evaluation

Cloud Service Analysis

Services Systems Manager
Simple Queue Service
Simple Notification Service
Security Token Service

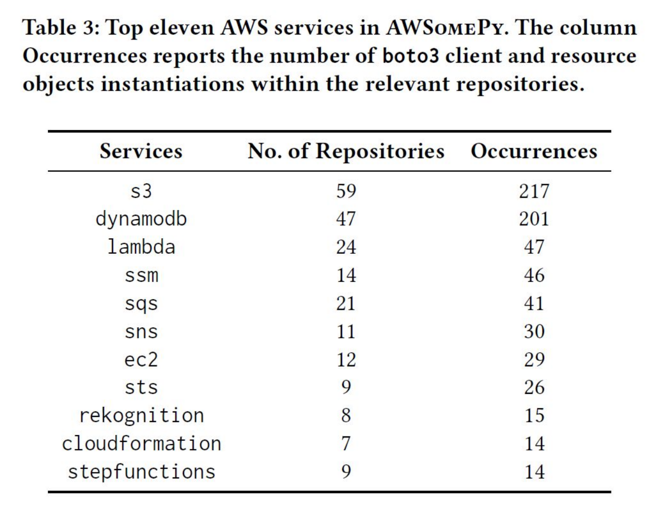


Table 3: Top eleven AWS services in AWSomePy. The column Occurrences reports the number of `boto3` client and resource objects instantiations within the relevant repositories.

Services	No. of Repositories	Occurrences
`s3`	59	217
`dynamodb`	47	201
`lambda`	24	47
`ssm`	14	46
`sqs`	21	41
`sns`	11	30
`ec2`	12	29
`sts`	9	26
`rekognition`	8	15
`cloudformation`	7	14
`stepfunctions`	9	14

Experimental Evaluation

Cloud API Usage Analysis

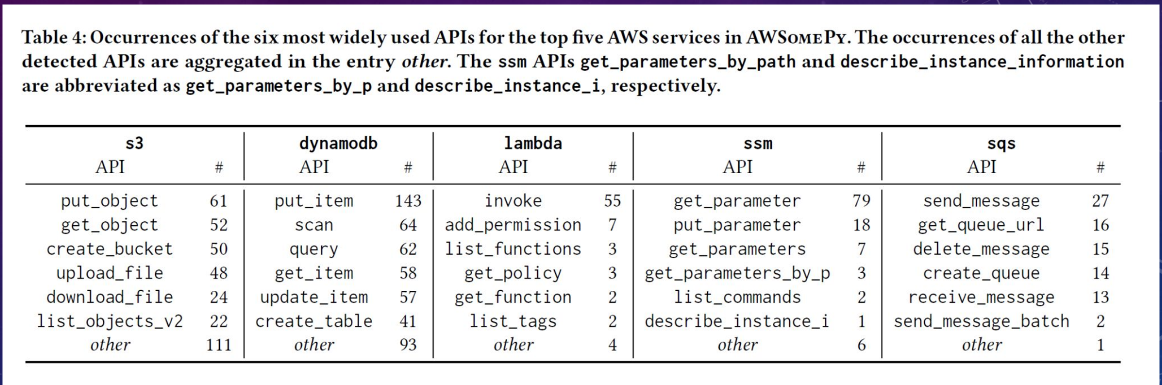
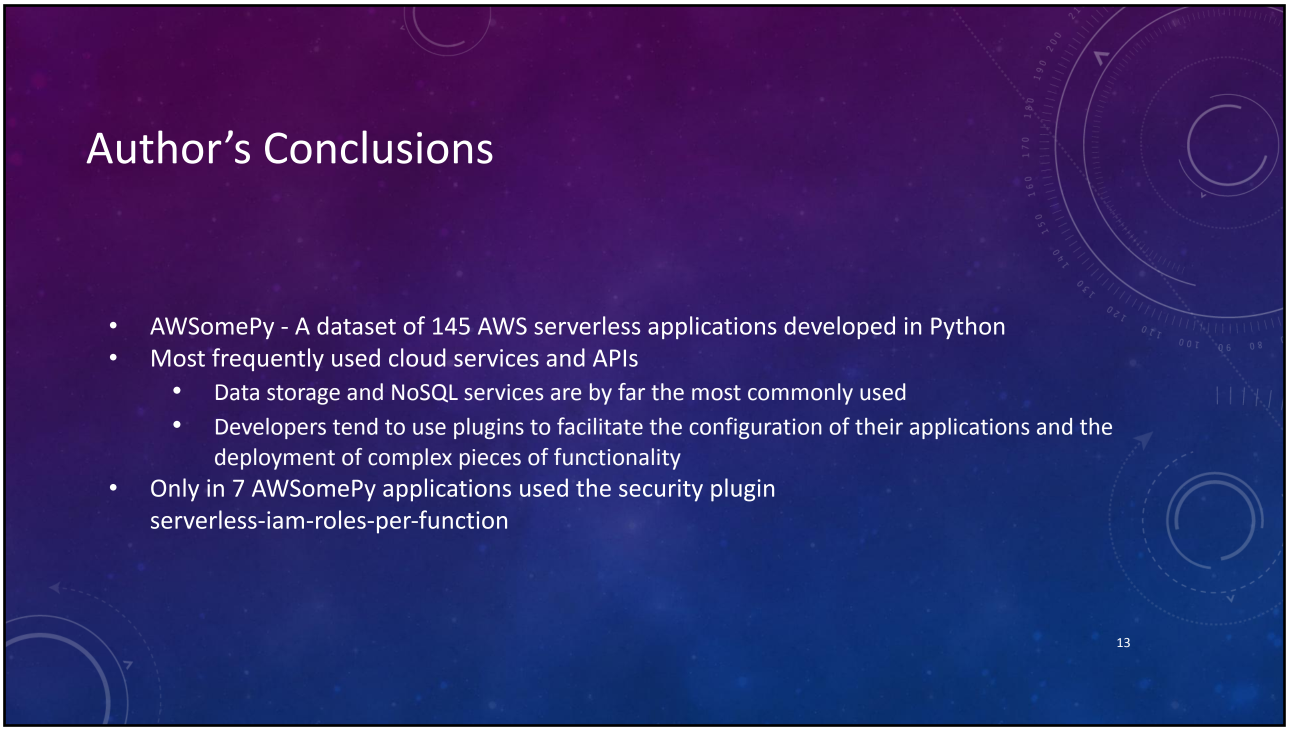


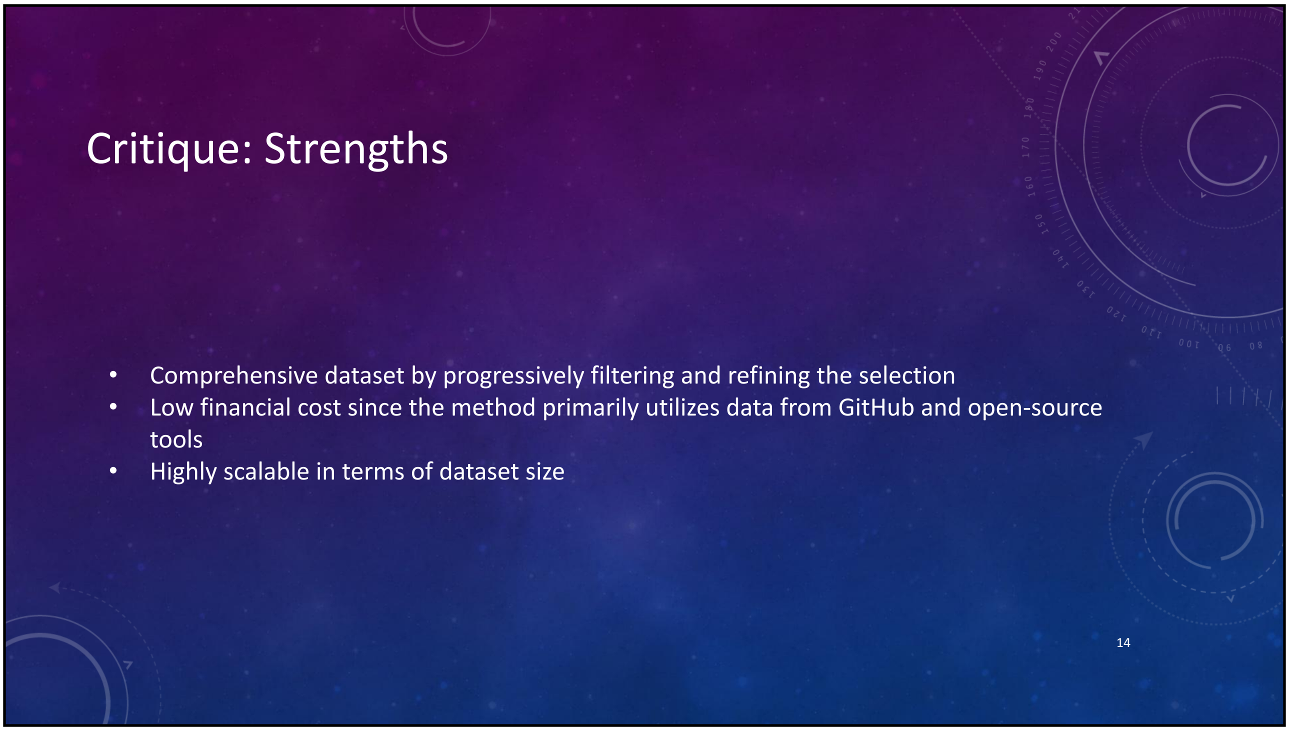
Table 4: Occurrences of the six most widely used APIs for the top five AWS services in AWSomePy. The occurrences of all the other detected APIs are aggregated in the entry *other*. The `ssm` APIs `get_parameters_by_path` and `describe_instance_information` are abbreviated as `get_parameters_by_p` and `describe_instance_i`, respectively.

s3		**dynamodb**		**lambda**		**ssm**		**sqs**	
API	#	API	#	API	#	API	#	API	#
`put_object`	61	`put_item`	143	`invoke`	55	`get_parameter`	79	`send_message`	27
`get_object`	52	`scan`	64	`add_permission`	7	`put_parameter`	18	`get_queue_url`	16
`create_bucket`	50	`query`	62	`list_functions`	3	`get_parameters`	7	`delete_message`	15
`upload_file`	48	`get_item`	58	`get_policy`	3	`get_parameters_by_p`	3	`create_queue`	14
`download_file`	24	`update_item`	57	`get_function`	2	`list_commands`	2	`receive_message`	13
`list_objects_v2`	22	`create_table`	41	`list_tags`	2	`describe_instance_i`	1	`send_message_batch`	2
other	111	*other*	93	*other*	4	*other*	6	*other*	1



Author’s Conclusions

- AWSomePy - A dataset of 145 AWS serverless applications developed in Python
- Most frequently used cloud services and APIs
  - Data storage and NoSQL services are by far the most commonly used
  - Developers tend to use plugins to facilitate the configuration of their applications and the deployment of complex pieces of functionality
- Only in 7 AWSomePy applications used the security plugin serverless-iam-roles-per-function



Critique: Strengths

- Comprehensive dataset by progressively filtering and refining the selection
- Low financial cost since the method primarily utilizes data from GitHub and open-source tools
- Highly scalable in terms of dataset size

Critique: Weaknesses

- Might be time-intensive, especially in manual steps like repository metadata analysis and may become less feasible as the dataset grows.
- May not capture the full dynamism and variability inherent in serverless architectures, especially where configurations are not standard or involve multi-file setups.

Critique: Evaluation

- Graphs and tables in the paper are clear
- Detailed evaluation in its analysis of plugin usage and service/API interactions
- 'serverless-iam-roles-per-function' plugin will not cause too much security problems.
- The evaluation parts only made a general analysis with only few concrete conclusions or suggestions.

Identify GAPS

- Apply with languages other than python
- Automation for data extraction and data analysis
- Comprehensive security analysis

Questions

KEY CONTRIBUTIONS

- Creation of the AWSomePy Dataset
- Analysis of Serverless Applications
- Service and API Utilization Metrics
- Evaluation and Insights