Graph-based Cloud Resource Cleanup

Netanel Cohen and Anat Bremler-Barr
The Interdisciplinary Center
Herzliya, Israel

The Problem

- As time passes, organizations that use cloud computing accumulate unused resources such as VM instances, Storage volumes and Databases. These unused resources:
  - Raise the monthly cost on public clouds.
  - Reduce capacity and degrade performance on private clouds.
  - Impose an additional operational burden.
  - Add security concerns.

General Idea

- We propose Garbo, a system that enables cloud resource cleanup by:
  1. Receiving **Core Resources** (i.e. used resources with non-cloud dependencies) as input from the user.
  2. Automatically generating a directed graph with cloud **resources** as nodes and dependency **relations** (e.g. A VM using a Storage volume) as edges.
  3. Performing Mark & Sweep on the graph, using **Core Resources** as roots.
  4. Producing a report of unused resources.

Cloud Resources Graph

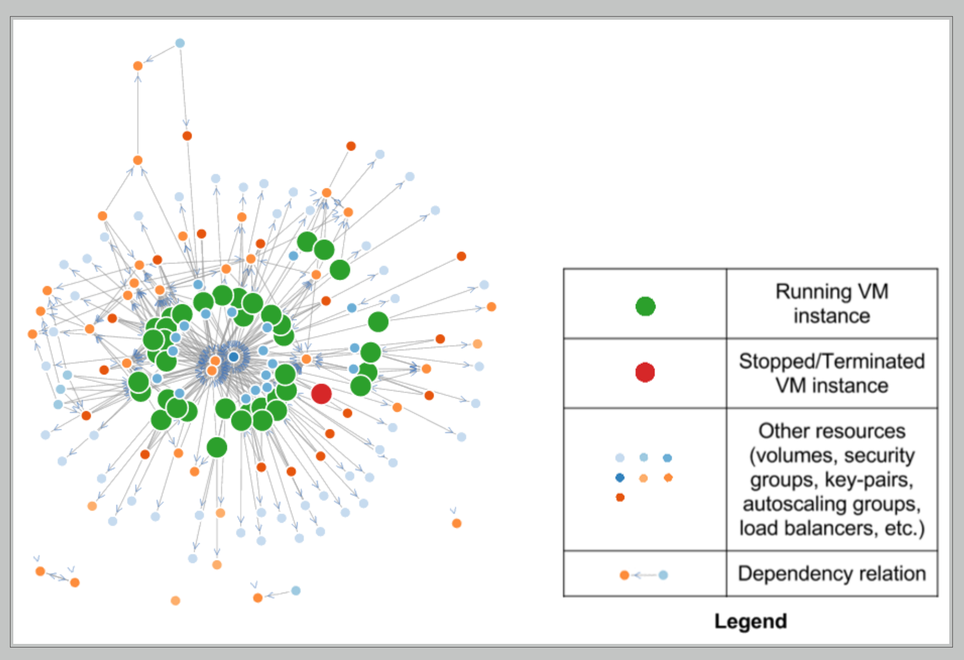

Figure 1: Cloud Resources Graph

Our Architecture

- Input of used **Core Resources**, e.g.
  - Web application's DNS record (Figure 3)
  - Batch Processing Autoscaling Group
- **Discovery Plugins** collect resources and relations from
  - Cloud API
  - Configuration Management API
  - CI/CD Tools API
- The system infers all used resources using the graph, and compiles a list of unused resources.

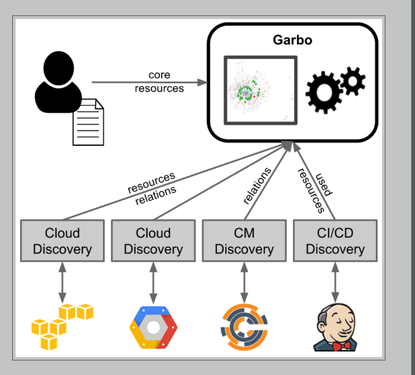

Figure 2: Architecture

Example: Core Resource in Web Application

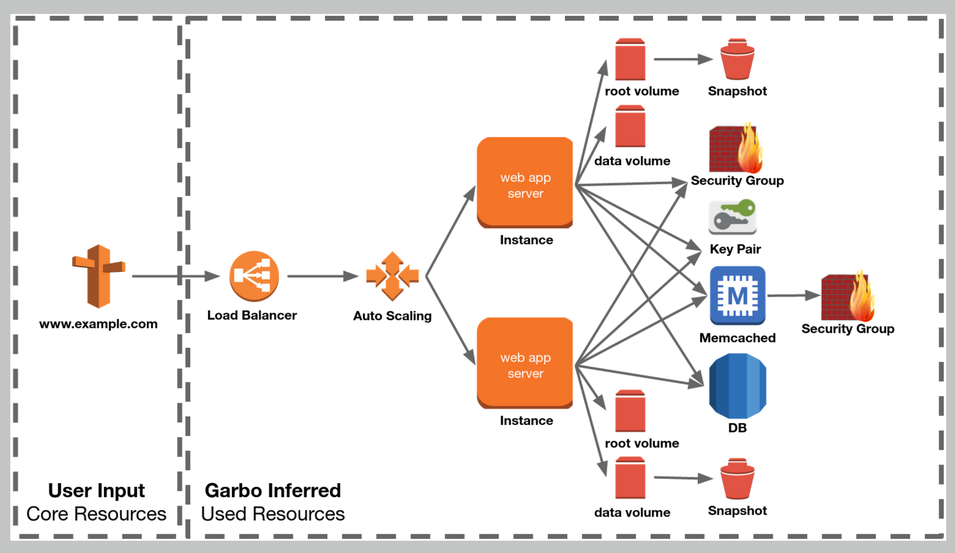

Figure 3: Core Resource in Web Application

Evaluation

- Current version implements AWS discovery
  - 11 Resource types, 18 Relation types
- Staging account of an anonymous company
  - 168 Resources, 401 Relations
  - 28 Core Resources, 8 Applications
- **Results:**
  - 14 Unused Resources (Figure 4)
  - 13 Verified by the System Administrator
  - 1 Default Cloud Resource (unused, but cannot be released)

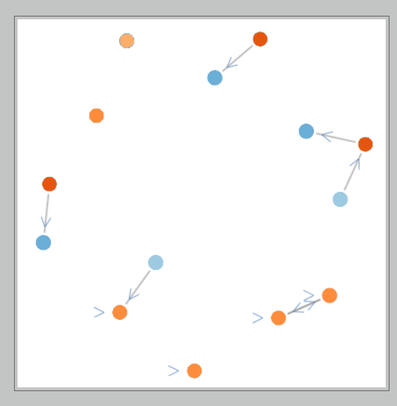

Figure 4: Unused resources output

Related Work

- Resource Cleanup
  - Poncho, Devoid et al 2013 - requires annotation per resource
  - Janitor Monkey, Netflix 2013 - requires rule set per resource type
- Other usages of a resource graph
  - Enterprise Topology Graphs, Binz et al 2012

Challenges & Future Work

- Dynamic cloud environments change rapidly
  - Asynchronous and inconsistent APIs
- Modeling resources and relations
  - Resource granularity
  - Relation directionality
- **Future Research**
  - Unique resource identification across multiple Discovery Plugins (Figure 5)
  - Detect Core Resources algorithmically
  - Online cleanup, using cloud logging (e.g. AWS Config, GCE Activity Logs)
  - Use the graph to detect failure domains

Challenge: Unique resource identification

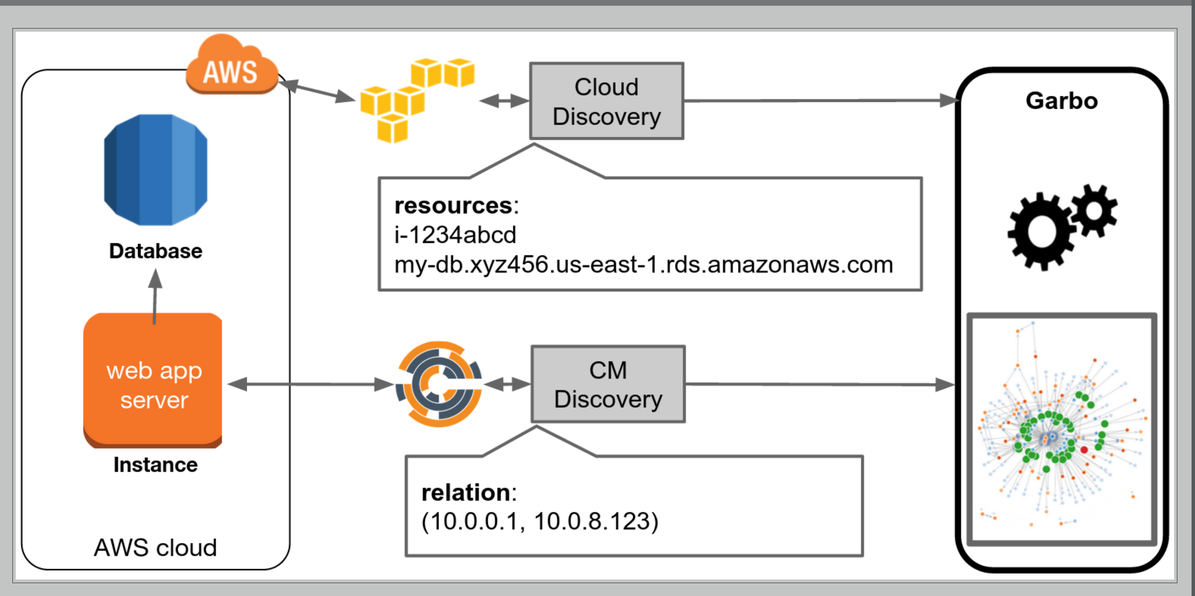

Figure 5: **Configuration Management Discovery Plugin** might identify resources using IP addresses, while **Cloud Discovery Plugin** will use cloud identifiers

Code

- Our code is available under MIT License at:
  - https://github.com/natict/garbo

Acknowledgments

- We are grateful to Avishai Ish-Shalom (Fewbytes) who provided helpful comments and suggestions regarding this work.
  This research was supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement №259085.

netanel.cohentzema@post.idc.ac.il

Garbo: Graph-based Cloud Resource Cleanup

Netanel Cohen Anat Bremler-Barr
The Interdisciplinary Center Herzliya, Israel
netanel.cohentzema@post.idc.ac.il bremler@idc.ac.il

1. Summary

Over the last few years, the number of organizations delivering their applications using public or private Cloud Computing has grown drastically. Each application is usually composed of many different resources, including: VM instances, storage volumes, databases, load balancers, and more.

Allocating additional resources is simple, and is usually accomplished by using dashboards, command-line utilities, cloud APIs, or cloud orchestration and automation tools.

However, the longer organizations use such cloud systems, the more unused resources they accumulate, for multiple reasons: Forgetting to release all resources associated with unused applications, Resources becoming irrelevant over time, Network errors while using the APIs and Software bugs.

Unused resources are a problem because they might raise the monthly cost on public clouds, reduce capacity and degrade performance on private clouds, impose an additional operational burden, and add security concerns.

In this poster, we propose Garbo, a system that given minimal user input emits a set of unused resources.

To do that, Garbo generates a dependency graph composed of cloud resources as nodes and their relations as edges. This resource information is acquired by a set of Discovery Plugins, which may use cloud APIs, configuration management APIs, or Continues Integration APIs. These allow the Garbo system to easily support additional resources residing in various public and private cloud providers.

User input to Garbo consists of a minimal set of Core Resources which are essentially used resources with non-cloud dependencies (eg. DNS record of a web application). Using these Core Resources as the roots of a Mark & Sweep process allows garbo to infer additional resources used by each application, and effectively detect the unused resources.

We implemented garbo with AWS discovery, evaluated it on the staging account of an anonymous company, and validated our results with their system administrator.

Submission to SoCC '15, August, 2015, Kohala Coast, HI, USA.

2. Related Work

While significant research efforts exist in cloud resource management [1], there is only preliminarily work on unused resource cleanup.

- Poncho [2], annotates instances in OpenStack and uses these annotations to manage them. The main drawbacks are the tedious annotation of many different instances, and the lack of support for other cloud resources.
- Janitor Monkey [3], implements a rule-based system for resource cleanup. Its main drawbacks are the need to implement a different set of rules for each resource type, and that it's intrinsically hard to detect unused resources with a circular dependency.
- Enterprise Topology Graphs [4] proposes a formal method to describe cloud resources and applications topology, but doesn't discuss how to construct them for an existing environment or suggests using them for resource cleanup.

Acknowledgments

We are grateful to Avishai Ish-Shalom (Fewbytes) who provided helpful comments and suggestions. This research was supported by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC Grant agreement No 259085.

References

[1] Jennings, Brendan, and Rolf Stadler. "Resource management in clouds: Survey and research challenges." Journal of Network and Systems Management (2014): 1-53.

[2] Devoid, Scott, Narayan Desai, and Lorin Hochstein. "Poncho: Enabling Smart Administration of Full Private Clouds." LISA. 2013.

[3] Fu, Michael, and Cory Bennett. "Janitor Monkey - Keeping the Cloud Tidy and Clean." The Netflix Tech Blog. Netflix, 13 Apr. 2013. Web. 25 June 2015.

[4] Binz, Tobias, et al. "Formalizing the cloud through enterprise topology graphs." Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012.