A Toolset for Detecting Containerized Application's Dependencies in CaaS Clouds

Pinchao Liu¹, Liting Hu¹, Haitu Xu¹, Zhiyuan Shi¹, Jason Liu¹, Qingyang Wang², Jai Dayal³, Yuzhe Tang⁴

¹Florida International University  ²Louisiana State University  ³Intel Corporation  ⁴Syracuse University

Motivation

Container as a Service (CaaS):
- A form of container-based virtualization
- Container engines, orchestration and the underlying computing resources are delivered to users as a service from a cloud provider

Opacity between cloud tenants and providers:
- Tenants have no insight into the underlying datacenter network topology, link and load
- Providers have no insight into the tenant's workloads

Opacity can lead to suboptimal container placement:
- Naively placing the heavily communicating containers across the racks can lead to unacceptable latencies
- Naively co-locating the containers that run interactive queries, and batch jobs while eagerly consuming any idling resources on the same servers can lead to unacceptable latencies

Objectives

Transparency and privacy-preserving:
- No code modification to the containers or applications
- Non-intrusive tool based on passive measurements

Lightweight:
- Network sniffer tools may bring heavy burdens to the management software
- Need to have negligible CPU and memory overheads

Actionable:
- Insights can help management software better co-locate container ensembles on underlying hosts

Data Collection and Correlation

Data Collection:
- CPU: CPU usage per second in percentage terms (%) → α
- Memory: Memory usage per second (MB) → β
- I/O: Packets transmitted per second (Kbyte/sec) → γ

Distance Identification:

\[
\text{Distance}(X_i, Y_i) = \frac{1}{\text{corr}_i} \quad \text{when} \quad \text{corr}_i > 0
\]

\[
\text{corr}_i = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{n} \sum_{i=1}^{n} x_i y_i - \frac{1}{n^2} \left( \sum_{i=1}^{n} x_i \right) \left( \sum_{i=1}^{n} y_i \right)
\]

Evaluation

ROC curve shows the overall accuracy area is 0.93, which is considered as excellent level.

Significant throughput improvement with container placement for RUBiS based on detected dependencies

Summary

- Hidden dependencies between containers belonging to the same applications can be revealed by monitoring their resource usage statistics at runtime
- One can design a lightweight black-box toolset to detect these dependencies based on passive measurement
- We achieved high accuracy when evaluating the approach with real-world containerized applications

Contact and Acknowledgments

The result from this work has been accepted for publication at IEEE Cloud 2018, to be held in July 2-7, 2018, at San Francisco, CA, USA.

Contacts:  Pinchao Liu (pliu002@cs.fiu.edu)
Liting Hu (lhu@cs.fiu.edu)
Jason Liu (jlux@cs.fiu.edu)

Website:  https://elves.cs.fiu.edu/
http://people.cis.fiu.edu/liux/research/

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-062-000002.