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

Pinchao Liu[1], Liting Hu[1], Hailu Xu[1], Zhiyuan Shi[1], Jason Liu[1], Qingyang Wang[2], Jai Dayal[3], Yuzhe Tang[4]

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

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

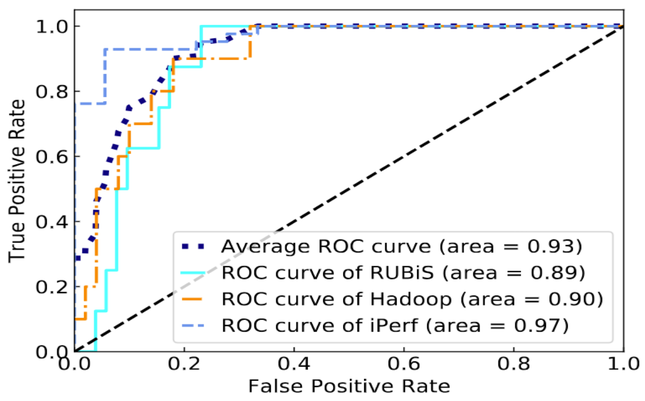
Distance Identification:

$$corr = \frac{\sum_{i=1}^{n}\left(\begin{pmatrix}X\alpha_i\\X\beta_i\\X\gamma_i\end{pmatrix}-\begin{pmatrix}X'\alpha_i\\X'\beta_i\\X'\gamma_i\end{pmatrix}\right)\left(\begin{pmatrix}Y\alpha_i\\Y\beta_i\\Y\gamma_i\end{pmatrix}-\begin{pmatrix}Y'\alpha_i\\Y'\beta_i\\Y'\gamma_i\end{pmatrix}\right)}{\sqrt{\sum_{i=1}^{n}\left(\begin{pmatrix}X\alpha_i\\X\beta_i\\X\gamma_i\end{pmatrix}-\begin{pmatrix}X'\alpha_i\\X'\beta_i\\X'\gamma_i\end{pmatrix}\right)^2}\sqrt{\sum_{i=1}^{n}\left(\begin{pmatrix}Y\alpha_i\\Y\beta_i\\Y\gamma_i\end{pmatrix}-\begin{pmatrix}Y'\alpha_i\\Y'\beta_i\\Y'\gamma_i\end{pmatrix}\right)^2}}$$

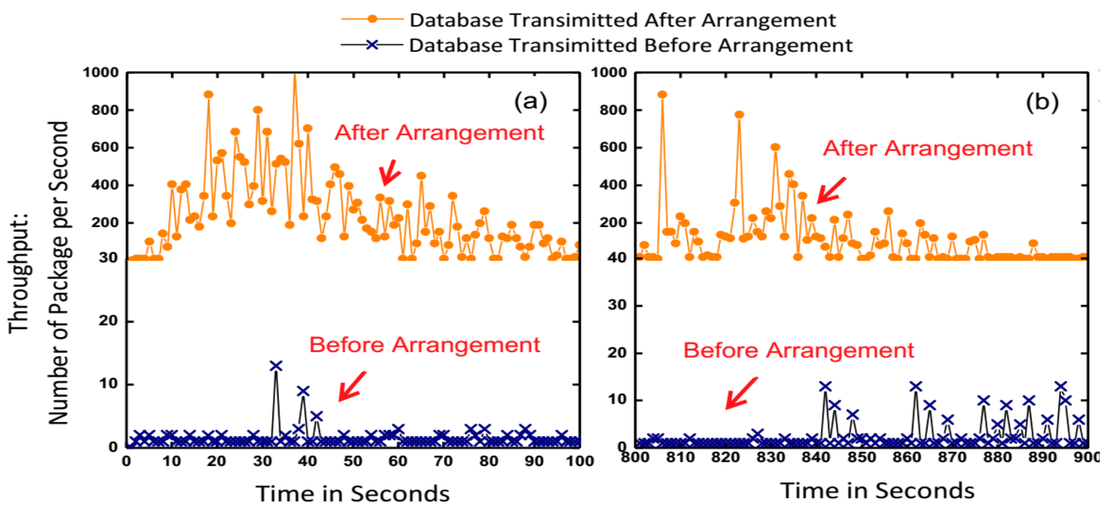
$$Distance(Xi, Yi) = \begin{cases}\frac{1}{corr_i} & , corr_i > 0\\ \infty, & corr_i = 0\end{cases}$$

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



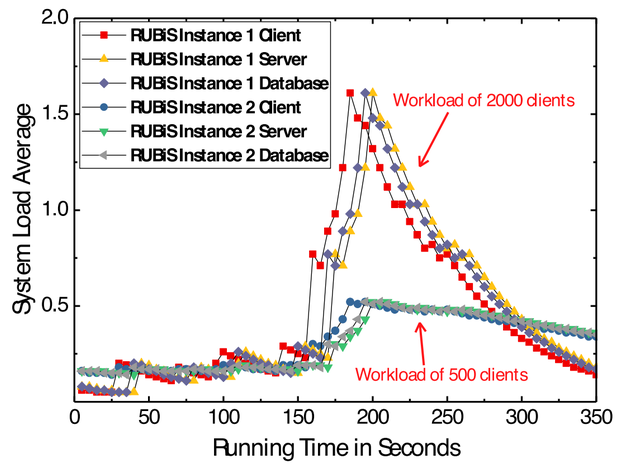
Objectives

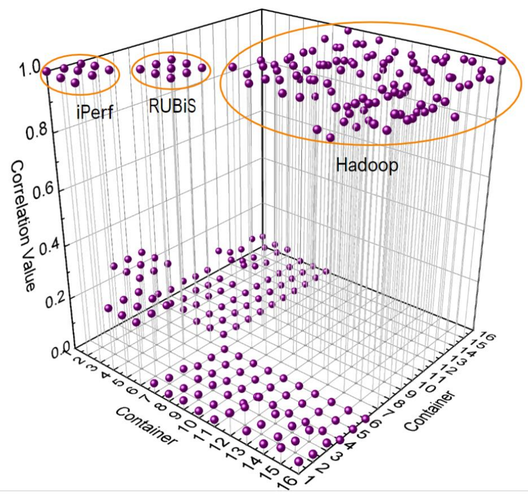
Transparency and privacy-preserving:
- No code modification to the containers or applications
- Non-intrusive tool based on passive measurements

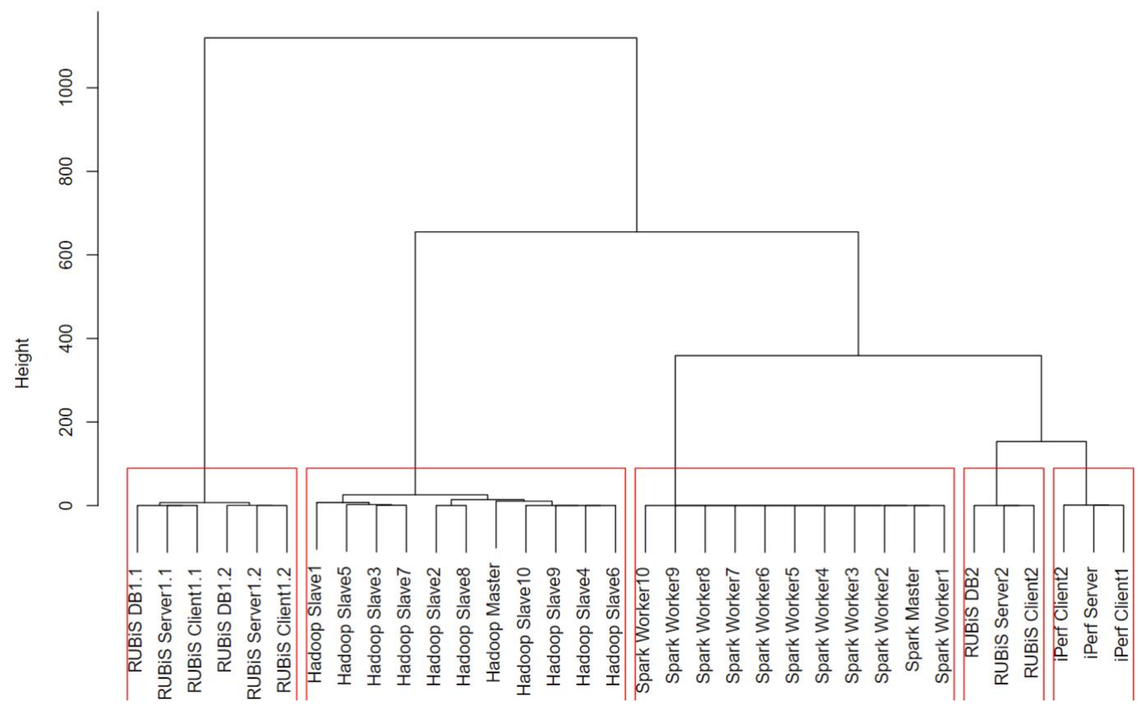
Lightweight:
- Network sniffer tools may bring heavy burdens to the management software
- Need to have negligible CPU and memory overheads

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

Hierarchical Clustering







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 light-weight 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 (liux@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.