# Container Memory Allocation Discrepancies:
## An Investigation on Memory Utilization Gaps for Container-Based Application Deployments (Extended Abstract)

Garrett Lahmann[1], Thom McCann[2], Wes Lloyd[3]

University of Washington
Institute of Technology,
Tacoma, Washington USA
[1]lahmanng, [3]wlloyd@uw.edu

[2]T-Mobile USA Inc.
Bellevue, Washington USA

***Abstract*— As cloud architectural platforms evolve, understanding how to maximize cost efficiency of application deployments is difficult without the ability to assess cost vs. performance tradeoffs of new technology platforms. With the advent of container-based computing, new opportunities for improving resource utilization efficiency have emerged. Compared to traditional cloud application deployments hosted on dedicated virtual machines (VMs), deployments to container clusters can save significant resources by aggregating application deployments to a shared pool of VMs. However, the degree of savings is often uncertain, and hobbled by excessive container resource allocation reflective of engineers' instincts to treat them as individual VMs. As practitioners are accustomed to performing application deployments to VMs, we are especially interested in understanding if VM resource allocations (e.g. CPU, RAM, disk) are appropriate for container deployments. In this research, we set out to analyze gaps between memory allocation and memory utilization for application deployments to container clusters.**

***Keywords Resource Management and Performance; Containerization; Application Profiling; IaaS; Multi-Tenancy;***

### I. INTRODUCTION

For many years now, practitioners have been migrating software applications to Infrastructure-as-a-Service (IaaS) cloud platforms. [1] Recently, application packaging and deployment has been revolutionized through the advent of application containers such as Docker and Rocket. Application containers leverage advancements of operating system containers such as LXC and OpenVZ to focus *specifically* on the deployment and hosting of individual application components with unique container instances. Cloud providers have also begun to offer container hosting services as an alternative to traditional IaaS VM hosting. Service examples include: Amazon Elastic Container Service (ECS), Azure Container Service (ACS), Azure Kubernetes Service (AKS), and the IBM Container Service. These services are backed by either vendor specific container orchestration framework such as Amazon ECS, or by an open source framework such as Kubernetes, Docker Swarm, or Apache Mesos/Marathon.

In the public cloud, users leverage container orchestration frameworks to create container clusters. These frameworks provide infrastructure management capabilities to enable users to aggregate hosting of multiple applications across shared clusters of VMs. The ultimate goal is to better leverage idle VM resources by moving away from the traditional model of deploying each application to separate cloud-based VMs. Moving to a shared resource model for application hosting is not without problems. VM container hosts must provide adequate resources with respect to CPU capacity, RAM, and disk space to support co-located application deployments. Additionally, resource isolation may become an issue as resource contention among co-located applications can lead to unexpected performance variation and/or degradation [2][3][4].

Unlike VMs, application deployments to containers such as Docker, do not duplicate RAM allocations for redundant operating system processes. They generally require less RAM than an equivalent operating system container or VM [5]. This fact is often overlooked by developers familiar with traditional VM based application deployments. While significant research has explored VM placement and resource allocation in public and private clouds, little research has considered how the agile nature of containers changes the equation of resource allocation for container deployments.

In this research, we investigate resource utilization of containerized application deployments to T-Mobile's in house Cloud Container Platform (CCP). CCP provides shared container clusters using Amazon EC2 VMs that leverage and extend upon the open source Apache Mesos/Marathon container orchestration framework to provide application hosting. We profile live application deployments to CCP to capture statistics including peak and current memory utilization, as well as allocation at the container level, and average memory utilization at the application level. We capture memory statistics: (1) for static deployments, (2) while stress testing applications with synthetic workloads, and (3) when varying container memory allocations to observe resulting performance implications. We explore memory allocation vs. utilization for hosting T-Mobile web applications with a variety of container configurations. Insights from our analysis are intended to support the development of performance models that will help to minimize over-provisioning of container resources through prediction of containerized application resource requirements.

#### A. *Research Questions*

For this research, we investigate the following research questions:

**RQ-1:** What gap exists, if any, between container memory allocations, and container memory utilization for container deployments to container clusters (e.g. Kubernetes, Apache Mesos/Marathon)? Is memory typically over-allocated?

**RQ-2:** From an organizational perspective, is memory over-allocation intentional, or the result of developer misjudgment of workload resource needs? How are memory allocation decisions made?

**RQ-3:** For observed instances of container memory overallocation, to what extent can memory allocations be safely reduced to match memory utilization before impacting application performance?

**RQ-4:** Is there a trend of memory over-allocation for specific application components hosted by containers? (e.g. redis, nginx, relational databases, application servers, microservices hosting, etc.)

Our workshop presentation will discuss findings on the following topics:

- Docker container memory allocation and utilization data from T-Mobile Cloud Container Platform application deployments
- Software architectures of our container-hosted applications
- Use of Linux /proc filesystem, collectd_docker, and the docker stats API to obtain memory utilization data/metrics
- The case for providing the ability to modify resource allocations of container deployments (e.g. Docker update) in container orchestration frameworks including what features are provided by existing frameworks (e.g. Kubernetes, Apache Mesos/Marathon), what features are lacking, and the potential for in-situ resource allocation changes to mitigate memory allocation challenges in real time

### *B. Contributions*

In this research, we investigate how container resource allocations are presently determined, and identify gaps between resource allocations and utilization of real-world application deployments to container clusters. **We argue that the best practices for container resource allocation should not simply be construed as the same as for VM resource allocation.** Given the agile nature of container deployments, their average lifetime, and the ease of dynamically adapting memory and CPU allocations, we argue that fine-grained container resource allocations are both feasible and desirable. By leveraging data from 27 application deployments at T-Mobile, we contribute a real-world case study that investigates issues of cloud resource allocation and management pertaining to containerization.

## II. Experimental Approach

As of early 2018, across four environments including production, staging, development, and performance lab, T-Mobile's CCP manages 1200-1800 docker containers deployed across 400-600 VM container hosts to support 27 application deployments at any given time. CCP's Docker containers are hosted on Amazon AWS EC2 instances sized from m4.2xlarge to m4.4xlarge. With 8 and 16 vCPUs, as well as 32 and 64GB of RAM respectively [2], these instances are currently provisioned with an average of 4 Docker containers due to the bottlenecking metric: allocated memory. It is estimated based on initial observations that the average number of containers per instance could be increased to as many as 8 given more stringent virtual memory allocation.

Metrics are collected every 30-seconds using a preinstalled collectd_docker plugin on each Docker host. Collectd_docker leverages the Docker Stats API for real time data collection at the container level. Collected metrics include: memory allocation, memory utilization, and peak usage. Capturing this data enables statistical analysis and modeling to support investigation of our research questions 1-4.

Figure 1 below depicts the percent of memory utilization vs allocation for 21 applications deployed to the CCP staging container cluster on EC2. The graph depicts average memory utilization for all containers of each application at an arbitrary point in time. **Average utilization per application in staging is just 4.64%, and per container 5.15%.**

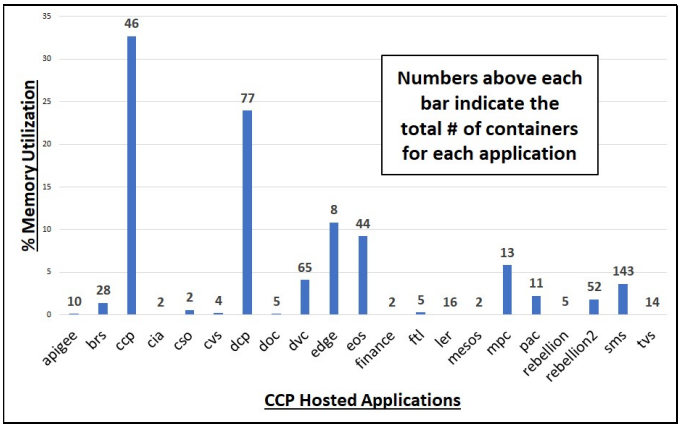


FIGURE 1: % MEMORY UTILIZATION FOR CCP APPLICATIONS

## REFERENCES


[1] Lloyd W, Pallickara S, David O, Lyon J, Arabi M, Rojas K. Migration of multi-tier applications to infrastructure-as-a-service clouds: An investigation using kernel-based virtual machines. Proc 12th IEEE/ACM Int. Conf. on Grid Computing (GRID 2011), Sept. 2011, pp. 137-144.

[2] J. Schad, J. Dittrich, J. Quiane-Ruiz, Runtime measurements in the cloud: observing, analyzing, and reducing variance, Proc. of the VLDB Endowment, v. 3, no.1-2, Singapore, Sept. 2010, pp. 460-471.

[3] Lloyd W, Pallickara S, David O, Arabi M, Rojas K. Mitigating Resource Contention and Heterogeneity in Public Clouds for Scientific Modeling Services. In Proc. 2017 IEEE Int. Conf. on Cloud Engineering (IC2E), Apr 2017, pp. 159-166.

[4] Xavier MG, Neves MV, Rossi FD, Ferreto TC, Lange T, De Rose CA. Performance evaluation of container-based virtualization for high performance computing environments. In 2013 21st Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing (PDP), Feb 2013, pp. 233-240.

[5] W. Felter, A. Ferreira, R. Rajamony and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers", in Performance Analysis of Systems and Software (ISPASS), 2015 IEEE Int. Symposium on, Philadelphia, PA, USA, 29-31 March 2015.