컨테이너 환경에서 과학 워크플로우를 위한 재구성 기반의 메모리 할당 기법

아두푸 테오도라 ○, 최지은, 김윤희*
숙명여자대학교 컴퓨터과학과
{theodora, jechoi1205, yulan}@sookmyung.ac.kr

A Reconfiguration-based Memory Allocation Method for Scientific Workflows in Containers

Theodora Adufu, Jieun Choi, Yoonhee Kim
Sookmyung Women's University

요 약

A key challenge during scheduling of scientific HPC applications remains the over-provisioning of virtual resources for optimal performance during execution. Even though this guarantees the peak performance of scientific application in virtualized environments, it results in increased amounts of idle resources which are unavailable for use by other applications. We propose a memory resource reconfiguration approach which allows the quick release of idle memory resources to new applications in OS-level virtualized systems, based on the application's resource usage pattern profile data. Our proposed approach fine-tunes memory allocation to containers at each stage of the workflow's execution life-cycle and hence improves overall memory resource utilization.

1. Introduction

Virtualization technologies [1], [2], [3], [4] serve as efficient solutions for the execution of scientific HPC applications with increasing resource demands through dynamic provisioning of shared resources. However, users tend to over-provision memory resources to applications to guarantee adequate resource availability during rare periods of high resource demands [5]. A key challenge remains the variation in memory resource usage during the execution life-cycle of HPC scientific workflows. Consequently, this increases the operational costs of deployment [6].

We propose a resource allocation method which considers the "bursty" memory usage pattern of applications and improves memory usage efficiency of a container throughout the execution life-cycles of an application by allocating memory resources to workflows based on the maximum required resources per task of the workflow to maximize the overall utilization. The proposed method is also able to reduce the number of jobs in a queue by releasing idle memory resources from containers to small jobs during periods of low memory usage. We use Docker containers to mitigate the platform overheads [7], [8], [9] of HPV technologies.

In this paper, related works are presented in Section 2 and we describe the proposed method in Section 3. In section 4, we give the details of the experiment with corresponding results and conclude in Section 5.

2. Related Works

Few research works aim to maximize resource utilization by employing dynamic reconfiguration methods during resource allocation in virtualized environments. Some works have focused on application-aware schemes which consider the characteristics of the application before allocating resources in an attempt to satisfy users' Service Level Agreements (SLAs).

The "try-before-buy"[10] resource allocation method aims to reduce contention among shared resources. Kundu et al [11] continuously make changes to resource allocations across multiple iterations to facilitate redistribution of resources. DejaVu [12] reuses previous resource allocations for each class of clustered resources. These methods however, do not consider the changing resource

*교신저자

demands of tasks within the application to maximize resource usage efficiency per application.

3. Reconfiguration-based allocation Method

The ultimate goal of the proposed method, is to maximize the use of idle resources of applications with "bursty" resource demands throughout their execution life-cycle without compromising their performance. For each group of tasks, our method first uses the resource usage and execution time (ET) profiles to estimate the memory requirement (MEst). Next, we maximize resource usage efficiency by assigning required memory resources dynamically. For modules with significantly lower resource demands than the previous, our method is able to release idle resources to other applications. The proposed method is thus able to allocate resources to optimize memory usage efficiency per container.

4. Experiments and Results

We deploy Ubuntu 14.04-based container images, on a CentOS 7-based Intel(R) Core (TM) i7 CPU 950 @ 3.07GHz server machine with 4 GB RAM. The target application, Montage GALFA [13] scientific workflow is pre-installed in the container.

Montage is a high performance Astronomical Image Mosaic Engine for creating composite FITS mosaics using multiple astronomical images. We obtain a set of five data cubes and aggregate them into a mosaic in three major stages: data cube shrinking, re-projection and aggregation.

Montage GALFA has 5 major groups of tasks or modules, each with different memory resource demands and execution times as seen in table 1. Using Valgrind profiling tool [14] and executable bash scripts, we obtain the memory usage and execution time data of each module of Montage GALFA as seen in table 1. From the profiles, we identify the modules for which memory resources will be reconfigured during execution as mShrink (mS), mMakeHdr (mMH), mProjectCube (mPC), mAddCube (mAC) and, reconfigure the container's memory resources accordingly.

We compare the memory released (MRel) for the case when reconfiguration is not performed, (NR: No reconfiguration) and for the case when there is reconfiguration (R: With reconfiguration) per module in our workflow, in one container.

Table 1. Memory allocation, Released Memory and Memory usage Efficiency for Reconfiguration and No Reconfiguration methods

Module	ET, secs	MEst (MB)	NR: Alloc (MB)	NR: Eff (%)	R: Alloc (MB)	R: MRel (MB)	R: Eff (%)
Initial	0	0	2414	0	0	2414	0
mS	234	11.7	2414	0.48	12	-2402	97.3
mIT	0.29	11.7	2414	0.49	12	0	97.6
mMH	0.05	16.8	2414	0.70	17	5	98.8
mPC	238	2413	2414	99.9	2414	2397	99.9
mIT	0.23	11.7	2414	0.49	18	0	61.6
mAC	859	1682	2414	69.7	1683	-731	99.9

The initial memory allocation for both methods is 2414MB and the memory released per module is shown in Table 1. R method releases memory of up to 2402MB for use by other applications with execution times shorter than 234 seconds. NR however does not reconfigure resources throughout execution and thus does not maximize memory usage.

We calculate the memory usage efficiency per module using equation 1 and observe that as the total amount of memory freed by using our approach increases there is better memory usage per container, eliminating waste introduced by memory resource over-provisioning.

$$\text{Eff}(\%) = \frac{\text{Memory Usage}}{\text{Memory Allocation}} \times 100 \qquad (1)$$

From the results, the highest memory efficiency (Eff) with reconfiguration, R, is 99.99% during the execution of mAddCube. For mProjectCube, the memory usage efficiency for both methods is the same at 99.98% since at this point, 2414MB is allocated to the container. Relatively, our approach is able to improve the memory utilization of the container by up to 96% for periods of low resource usage.

4. Conclusion and Future Works

We present a reconfiguration-based memory allocation method that maximizes the memory usage efficiency of containers. Our method uses memory usage profile data to plan the allocation of memory resources throughout the execution life-cycle in a manner that ensures that the required amount of memory is allocated per module for applications with

"bursty" resource demands. Results show that with our method, applications with "bursty" resource demands are able to release idle resources to other applications.

In the future we will expand the proposed method to reduce job queuing time of incoming tasks and the number of tasks in a queue through dynamic resource reconfiguration and provisioning schemes. Also, we will expand our resource allocation approach to support other computing resources such as disks, and cache memory.

References

[1] Nimbus, http://www.nimbusproject.org/
[2] OpenStack, http://www.OpenStack.org
[3] Amazon EC2, http://aws.amazon.com
[4] Docker, http://www.docker.com
[5] Kalyvianaki et al, "Ad Hoc Mobile Wireless Networks: Protocols and Systems", Prentice Hall PTR, New Jersey, 2002.
[6] Eun-Kyu Lee, "Cooperative Virtual Data Center: Sharing Data and Resources among Multiple Computing Entities", International Journal of Software Engineering and Its Applications Vol. 9, No. 11 (2015), pp. 137-152
[7] Adufu et al, "Is container-based technology a winner for high performance scientific applications?", Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific
[8] Memari et al, "Towards virtual honeynet based on LXC virtualization" Region 10 Symposium, 2014 IEEE, pp. 496-501
[9] Xavier et al, "Performance Evaluation of Container-based Virtualization for High Performance Computing Environments" Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, 2013, pp. 233-240
[10] Yu et al., "Resource Allocation in Virtualized Systems Based on Try-Before-Buy Approach", Published in 2nd International conference on Challenges in Environmental Science and Computer Engineering, 2011, pp 193-199.
[11] Hong et al, "Application-aware Resource Allocation for SDN-based Cloud Datacenters", Proc. Int'l. Conf. Cloud Computing and Big Data 2013, Fuzhou, China, Dec. 2013.
[12] Vasic et al, "DejaVu: Accelerating Resource Allocation in Virtualized Environments", ASPLOS'12, March 3-7, 2012, London, England, UK
[13] Montage, http://montage.ipac.caltech.edu
[14] Valgrind, http://www.valgrind.org/

Acknowledgement

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2015M 3C 4A7065646r)