Modeling, Characterising and Scheduling Applications in Kubernetes

Víctor Medel[1], Unai Arronategui[1], José Ángel Bañares[1], Rafael Tolosana[1], and Omer Rana[2]

[1] Aragón Institute of Engineering Research (I3A)
University of Zaragoza, Zaragoza, Spain
[2] School of Computer Science & Informatics
Cardiff University, UK

Abstract. The simplification of resource management for container is one of the most important services of Kubernetes. However, the simplification of distributed provisioning and scheduling decisions can impact significantly in cost outcomes. From an economic point of view, the most important factor to consider in container management is performance interference among containers executing in the same node. We propose a model driven approach to improve resource usage in overall deployment of applications. Petri Net models, a Confirmatory Factor Analysis (CFA)-based model and a regression model allows to predict performance degradation of the execution of containers in applications. Time series indices can provide an accurate enough characterisation of the performance variations in the execution lifetime of applications. These indices can be used in new scheduling strategies to reduce the number of resources used in shared cloud environments as Kubernetes.

Keywords: Modeling · Petri Nets · Confirmatory Factor Analysis · Interference · Scheduling · Containers.

1 Introduction

Faster start up times and fewer required resources are the main advantages of containers compared to traditional virtual machine technologies. These have result in a higher participation of container technology in private and public clouds. Kubernetes has become the facto standard for distributed execution of applications in containers. Deployment and scheduling of applications on shared Kubernetes platforms remains activities that can be improved also in their economic angle. However, it can require a substantial effort, due to the complexity of the cloud distributed infrastructures. This paper aims to enhance a modeling path to improve these aspects of application execution to reduce resource requirements and, therefore, cost.

Our approach is the use of formal models with a twofold goal: (1) a better management of the growing complexity of current systems; (2) a high quality of the implementation reducing the time to market. This poster presents in a cohesive way our works related with the modelling, characterisation of applications and scheduling in Kubernetes.

2 Models and Methodology

In [3, 5] we presented a High Level Petri Net (reference net) based performance and management model for Kubernetes, identifying different operational states that may be associated with a "pod" and container in this system. The model is an executable specification that can be used for performance evaluation. A quantitative analysis can be conducted by a performance-oriented interpretation of the model such as throughput, utilisation rates, or queue lengths, from which is possible compute rewards functions [6].

The Reference net formalism is a special class of high level Petri net (adhered to the Nets-within-Nets paradigm). The hierarchical construction of the model allows to follow a top-down approach incorporating more details in the lower levels. However, the construction of a complete model with all the details can be an impossible task when there are a big number of factors that can affect the system's behaviour, and there is not a clear relationship between them. In this case, the usual solution for incorporating the observed behavior to the model is to annotate the model with deterministic time, probability distributions, or functions obtained from the monitoring data acquired from benchmarking. This approach can capture the whole behavior of the computational resources, and, therefore, a more precise performance analysis can be obtained. This is the case of the modelling and characterisation of applications interferences in container deployments.

Performance degradation of containers running in the same machine can be observed when resources needed for one container are used by another one. The performance loss produced by the simultaneous execution of two containers on the same host is the measure of the interference between both containers. Also, this interference is time dependent, as resource requirements vary during execution of applications.

We consider several *sources of interference* rooted on physical resources hosting the container :

- *Network usage*: all containers on a node share network access, thus they can disturb each other
- *CPU usage*: a reservation system to share the CPU proportionally is applied in most container management systems if there is contention.
- *I/O file system access*: it has similar sharing behavior as the network.
- *Cache Memory* and Memory bandwidth: Containers can provoke cache misses to others containers running in the same host, degrading memory bandwidth.

We propose a *methodology* to estimate the interferences and to obtain functions to annotate our Petri net based performance model. It consists of different steps to estimate the execution time of an application when it's co-scheduled with another one. First, the interference profile of an application can be obtained following a process where the timed interference indices are modeled using Confirmatory Factor Analysis (CFA) [1]. This model is based on the definition of human-comprehensible indices to represent resource usage. These indices are computed from data sets obtained from experiments on resource usage from different benchmark applications and are expressed as time series to show the evolution of resource usage over time.

To validate our approach we executed different applications inside a container to generate a number of different jobs, each of which represents the application executed with different input parameters. These applications represent different profiles characterized by the intensive use of a certain type of resource. The objective is to get a dataset which captures the variations of the metrics to build meaningful indices. The high correlation between the observed variables avoids using them as raw values to describe an application and to do further analysis. We follow the approach of reducing all observed variables of resource usage to four human-comprehensible indices to represent resource usage over time: *CPU usage*, *Memory page fault*, *Memory hierarchy usage* and *Intensity of memory hierarchy usage*, and characterising resource usage of applications over time with these interference indices.

Afterwards, we measure execution time of this application in time intervals while simultaneously running benchmark applications on the same machine. Then a regression model can be defined from the first two steps for each application to obtain a interference linear function that models the application. Finally, these linear functions are used to estimate interference between two application whose interference linear functions are known.

3 Interference-aware Scheduling

When the number of tasks to schedule, at the same time, is greater than the amount of available computers in a distributed infrastruture, interference-aware scheduling is a policy that aims to minimize the performance degradation of these tasks, as explained in the previous section. The goal is to schedule, in the same machine, the tasks whose simultaneous execution produce the less performance degradation to each other.

In [4], we showed how the default Kubernetes scheduler was not suited to avoid performance degradation. Also, it was showed how a simple yet effective policy could reduce resource contention. In this work, we proposed a simple scheduling technique based on the characterisation of applications. The idea is that clients, or developers, provide informal information about the resource most intensively used by the application, and the scheduler uses that information to allocate the applications using the same resource in different machines. In our experiments, we achieved about a 20 percent improvement in the execution time of a simple scenario compared with the default Kubernetes non-deterministic scheduler. But it was a coarse grained approach that didn't take into account the variable requirements of resources during the execution of many applications, particularly long running applications such as services.

In Paragon [2], authors use a collaborative filtering algorithm to determine the influence of several sources of interference and propose an interference-aware scheduler. However, the main novelty of our approach compared with Paragon is that they considers interference remains constant over time.

4 Conclusions and Future work

This paper has presented a model driven approach to reduce costs linked to resource management strategies with containers in Kubernetes, using Petri Net Models, a Confirmatory Factor Analysis (CFA)-based model and a regression model. Different methodologies applied with these models allow to predict resource usages of applications. Time series indices provide a characterisation of performance variations of applications that can contribute to enhance scheduling policies in containers platforms as Kubernetes.

As future work, in addition to the scheduler based on interference functions, we will use the petri net model with these annotations to perform different performance and cost analyzes.

Acknowledgments This work was co-financed by the Aragonese Government and the European Regional Development Fund "Construyendo Europa desde Aragón" (COSMOS research group, ref. T35_17D); and by the Spanish program "Programa estatal del Generación de Conocimiento y Fortalecimiento Científico y Tecnológico del Sistema de I+D+i ", project PGC2018-099815-B-100.

References

1. Brown, T.A.: Confirmatory factor analysis for applied research. Guilford Press (2015)
2. Delimitrou, C., Kozyrakis, C.: Paragon: Qos-aware scheduling for heterogeneous datacenters. In: ACM SIGPLAN Notices. vol. 48, pp. 77–88. ACM (2013)
3. Medel, V., Rana, O., Bañares, J.a., Arronategui, U.: Modelling performance & resource management in kubernetes. In: Proceedings of the 9th International Conference on Utility and Cloud Computing. pp. 257–262. UCC '16, ACM, New York, NY, USA (2016)
4. Medel, V., Tolón, C., Arronategui, U., Tolosana-Calasanz, R., Bañares, J.Á., Rana, O.F.: Client-side scheduling based on application characterization on kubernetes. In: International Conference on the Economics of Grids, Clouds, Systems, and Services. pp. 162–176. Springer (2017)
5. Medel, V., Tolosana-Calasanz, R., Bañares, J.Á., Arronategui, U., Rana, O.F.: Characterising resource management performance in kubernetes. Computers & Electrical Engineering 68, 286–297 (2018)
6. Tolosana-Calasanz, R., Bañares, J.Á., Pham, C., Rana, O.F.: Resource management for bursty streams on multi-tenancy cloud environments. Future Generation Comp. Syst. 55, 444–459 (2016)