Towards distributed, fair, deadline-driven resource allocation for Cloudlets

Stratos Dimopoulos
stratos@cs.ucsb.edu
University of California, Santa Barbara

Chandra Krintz
ckrintz@cs.ucsb.edu
University of California, Santa Barbara

Rich Wolski
rich@cs.ucsb.edu
University of California, Santa Barbara

ABSTRACT

In this paper we present our vision for a two-level, distributed resource allocator that preserves fairness and satisfies deadlines of low latency workloads in a multi-cloudlet environment with offloading support. We analyze the opportunities and challenges that offloading and the multi-cloud environment impose and we suggest the changes required to a fair-preserving and deadline-driven resource allocator originally designed for resource-constrained environments.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**.

KEYWORDS

cloudlet, offloading, deadlines, fairness, distributed scheduling

ACM Reference Format:
Stratos Dimopoulos, Chandra Krintz, and Rich Wolski. 2019. Towards distributed, fair, deadline-driven resource allocation for Cloudlets. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mobile Cloud Computing (MCC) [7] allows mobile devices such as smart phones, tablets, and wearables to offload processing and data storage to the cloud [14] in order to mitigate their limited computational, storage, and power resources. However, interactive mobile applications like real-time multimedia, gaming, augmented-reality, and location-aware and guidance services, generate data streams that not only demand significant computational resources but they also require low latency processing in order to provide acceptable user experience. Offloading such services to the remote cloud data center would cause significant service degradation.

To provide higher computation resources with lower latency, processing can be done closer to where the data is produced in edge or fog computing clouds [2, 8, 18] and smaller "data-centers in-a-box" called cloudlets [17]. Unlike remote cloud data-centers though, resources on cloudlets are very limited. Therefore, significant research is focused on addressing the problem of choosing to offload between clouds and cloudlets taking into consideration the battery, bandwidth, and computational requirements [1, 3, 6, 9, 11, 13].

To allow low-latency processing, we have designed a predictive resource allocation mechanism based on admission control that can also adapt to changing cloud conditions in order to satisfy application deadlines and preserve fairness in resource-constrained cloud environments similar to those found in cloudlets. This allocator called *Justice* [5] is an application-agnostic framework for processing a variety of data analytics workloads. It employs a coarse-grained predictive mechanism that imposes very little overhead, making it ideal for resource-limited environments. We have evaluated *Justice* with production workloads and find that it enables fairness, deadline satisfaction, and efficient resource utilization when resources are highly constrained.

In this work, we explore the design of a resource allocator that is suitable for cloudlets, which can offload processing to remote clouds or cloudlets when necessary. We suggest a two-level, distributed allocation mechanism for multi-cloudlet environments that uses cloudlet and remote cloud coordination to improve offloading decisions, satisfy job deadlines, and to preserve fairness both locally (cloudlet) and globally (multi-cloudlet deployment).

It differentiates its policy across diverse application classes and exploits application-specific characteristics such as pre-emption and execution patterns, when/if available, to facilitate finer-grained prediction and adaptability. Based on historical job statistics, local and remote cloudlet workload information, the allocator determines which jobs should run on the current cloudlet and which should be offloaded on the remote cloud or on a neighbor cloudlet. We plan to evaluate our mechanism with cloudlet specific-workloads (compute intensive and latency sensitive) and a mixture of soft and hard deadlines corresponding to the different types of latency sensitive mobile applications.

2 BACKGROUND AND RELATED WORK

Justice [5] is an allocator designed for the resource-constrained settings of a cloudlet with the dual goal of preserving fairness and satisfying job deadlines for multi-analytics batch workloads. Justice does so by using an adaptive prediction technique based on historical job execution times to estimate the minimum number of CPUs a job requires to meet its deadline "just-in-time". It utilizes cloud resources efficiently by applying admission control and proactively denying service to jobs that cannot meet their deadlines based on the cloud conditions.

Figure 1 depicts the three concurrent operations of *Justice*. To estimate the resources required to satisfy job deadlines (used to guide both admission control and resource allocation), *Justice* employs an online statistical model from the job tracking operation. This

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

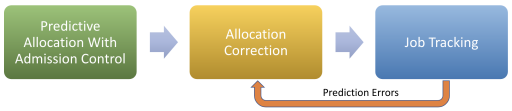

Figure 1: *Justice*'s predictive resource allocation and admission control with error correction based on online job tracking and a Kalman filtering feedback loop.

operation then feeds its estimation error to a Kalman filter and uses the filter's output to further correct future allocation predictions.

There are many aspects of the *Justice* design that make it suitable for resource allocation in cloudlets. First, *Justice* is able to utilize scarce resources more efficiently than other fair-share allocators. It achieves this in two ways. Firstly, it does not waste cloud resources to infeasible jobs and it does not over-provision resources to jobs with relaxed deadlines. Secondly, it incorporates coarse-grained prediction techniques that do not impose additional overheads to the cloud unlike offline simulations, sampling, or extensive online monitoring.

Next, *Justice* requires minimal information for the job (deadline and input size), doesn't increase scheduling latency, and its memory scales linearly to the points used for the desired history window specified. This window can be adapted depending on the prediction accuracy requirements and the scheduling latency overhead a cloudlet can tolerate. Finally, *Justice*'s prediction mechanism is framework-agnostic. This means that it does not depend on the internal structure or API of any specific processing engine and therefore it can be part of cloudlets that might be running different processing frameworks to potentially optimize for a specific application type.

Much past work has focused on the problem of load balancing between edge data centers [12, 15] and offloading work to remote clouds from cloudlets [3, 9, 11, 16]. Cardelini et al. [3] is using a game-theory approach to optimize the offloading decision, while Gelenbe et al. [9] formulates the choice between a local and a remote cloud as an optimization problem, taking into consideration energy and performance of servers. In [6], authors optimize offloading decisions and the allocation of computing resources, transmit power, and radio bandwidth focusing on user fairness and maximum tolerable delay. Other research determines the offloading decision based on the battery and bandwidth consumption [1] or volume of data need to be transferred and the computational resources required [13]. Authors in [16] want to decrease energy consumption and execution latency in a multi-cloudlet scenario by selecting the most appropriate neighboring cloudlet for each application type and also load balancing between cloudlets.

In our work, we focus in making the best decision after a job has already reached the cloudlet, in order to satisfy workloads in terms of their latency and allocate resources fairly between them. We want to use the limited resources of the cloudlet only for the workloads that absolutely need them in order to satisfy their latency requirements and send the workloads with more relaxed constraints to run on the virtually unlimited resources of a remote cloud or to less loaded neighboring cloudlets. We suggest a distributed scheduling scheme to coordinate resource allocation across cloudlets, inspired

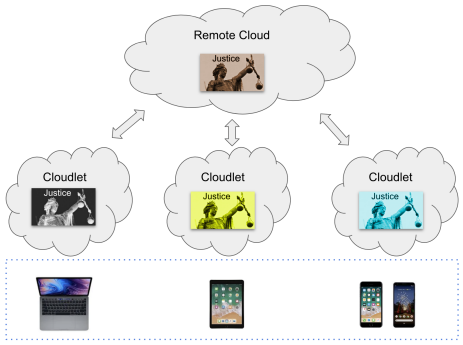

Figure 2: Distributed *Justice* in a multi-cloudlet environment

by two-level resource allocators for big data analytics like Apache Mesos [10] but specialized to overcome framework interference and fairness violation issues as these systems were not designed with resource-scarcity in mind [4]. Our work is also inspired by application specific cloudlets and multi-cloud environments [16], as this can be an extra incentive for collaboration between cloudlets.

3 DISTRIBUTING JUSTICE FOR MULTI-CLOUDLET ENVIRONMENTS

The ability to offload workloads from a cloudlet to a remote cloud and the potential for cooperation between multi-cloudlets, together create multiple scheduling and resource allocation opportunities for *Justice*. To exploit these opportunities, we augment the existing allocation mechanisms with support for offloading decisions to remote clouds and to neighboring cloudlets. *Justice* manages to keep cloudlet resources minimally utilized and proactively available for future workloads by predicting job resource requirements and allocating only the necessary resources so they can complete just before their deadlines.

Offloading workloads to remote clouds or other, less-busy cloudlets, adds extra parameters to *Justice*'s allocation mechanism. We expand *Justice*'s scope, from being specific just to the local cloudlet, to also consider statistical information from job executions that have been offloaded elsewhere (in the remote cloud, and to neighboring cloudlets) and utilization information of other cloudlets and remote-clouds.

We classify jobs based on their preemption characteristics and deadline types, as these aspects constrain our offloading and admission control decisions respectively. In particular, for jobs supporting pre-emption we can have more scheduling flexibility as we can offload them outside the cloudlet even after they have started running. Similarly, *Justice* can accept jobs with soft deadlines (i.e., the utility function of the job does not drop to zero after its deadline is exceeded), even when the confidence level of its prediction regarding meeting its deadline is lower.

To realize these new features, we expand the centralized architecture suitable for one cloudlet, to a distributed scheduling scheme capable of allocating resources in a group of neighboring cloudlets. *Justice* is ideal for that due to its simplicity and minimal computing requirements. Figure 2 shows a two-tier distributed architecture.

Justice instances run in both cloudlets and instances with increased responsibilities running in remote clouds (e.g., Amazon AWS, Microsoft Azure, Google Cloud etc). *Justice* instances running in remote clouds have a dual goal. Like all other instances, they are able to run to completion workloads offloaded to them. They are also able to act as brokers, responsible for communicating cloudlet utilization, historical runtime statistics, and coordination information across participating cloudlets (other *Justice* instances).

In addition to distributing *Justice*, so that we can provide fairness and satisfy latency requirements in multi-cloudlet environments, there are other aspects of such deployments that generate opportunities worth exploring. For scenarios in which each cloudlet is optimized to support only a specific type of application [16], we can quantify the runtime prediction improvement *Justice* achieves versus to its application-agnostic performance.

Justice is designed to accommodate for a variety of workloads. It achieves this by calculating globally applicable but also coarse-grained statistical information to predict job run times under changing cloudlet conditions. However, this can lead to prediction inaccuracies for some types of workloads. By considering workloads with similar characteristics, we can reduce this error. *Justice* achieves this by characterizing and exploiting job repetition and dependency information to improve its allocation decisions. Doing so is not efficient for cloudlets supporting generic workloads, as it requires expensive machine learning (i.e., clustering) to classify the workloads according to their runtime and resource requirement characteristics. However, it may be possible for application-specific, multi-cloudlet deployments. In such deployments, for example, one cloudlet might be optimized for augmented reality applications, while other cloudlets might be more suitable for mobile games, mobile health applications, numerical operations, file operations (searching, sorting) and so on. We expect that workloads of the similar applications share runtime and repetition characteristics that will allow *Justice*'s mechanism to be more accurate.

Along with the many opportunities for allocation improvement, a distributed scheme with multiple coordinating *Justice* instances, generates new research challenges in terms of preserving fairness, satisfying deadlines, and keeping scheduling latency low. Fairness can no longer been seen in one-dimension. Each instance of *Justice* running on a cloudlet has sufficient information to enforce fairness for its cloudlet. However, there is a second dimension of fairness; the one preserved across all collaborating cloudlets in a multi-cloud environment. To achieve this, *Justice* instances must collaborate to create a global fairness "snapshot" based on partial local knowledge. We must also consider the tradeoff between the amount of statistical information from remote clouds and cloudlets that each cloudlet stores locally versus the amount of information exchanged between the distributed *Justice* instances. Lastly, we must expand *Justice* to consider placement constrains (e.g., GPU versus CPU or specific operating system requirements) and preserve fairness across different job priorities to evaluate its performance in more realistic situations corresponding to diverse MCC workloads and mobile user service levels. For example, workloads belonging to a higher service level should be executed in the least utilized cloudlet (between two cloudlets with the same proximity to the mobile user) and this is a decision that can be augmented by utilization information we keep on the remote cloud.

4 CONCLUSIONS

We present our vision for a distributed two-level scheduler based on *Justice* [5] suitable for multi-cloudlet environments with resource offloading to remote public clouds. We discuss our motivation and analyze the new feature set that *Justice* must support to preserve fairness and satisfy job latency requirements in the context of a multi-cloudlet and public-cloud hybrid environments and MCC workloads.

ACKNOWLEDGMENTS

This work is funded in part by NSF (CNS-1703560, CCF-1539586, ACI-1541215)

REFERENCES

[1] Marco V Barbera, Sokol Kosta, Alessandro Mei, and Julinda Stefa. 2013. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *2013 Proceedings Ieee Infocom*. IEEE, 1285–1293.

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 13–16.

[3] Valeria Cardellini, Vittoria De Nitto Personé, Valerio Di Valerio, Francisco Facchinei, Vincenzo Grassi, Francesco Lo Presti, and Veronica Piccialli. 2016. A game-theoretic approach to computation offloading in mobile cloud computing. *Mathematical Programming* 157, 2 (2016), 421–449.

[4] Stratos Dimopoulos, Chandra Krintz, and Rich Wolski. 2016. Big Data Framework Interference In Restricted Private Cloud Settings. In *IEEE International Conference on Big Data. IEEE*.

[5] Stratos Dimopoulos, Chandra Krintz, and Rich Wolski. 2017. Justice: A Deadline-aware, Fair-share Resource Allocator for Implementing Multi-analytics. In *Cluster Computing (CLUSTER), 2017 IEEE International Conference on*. IEEE, 233–244.

[6] Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. 2018. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications* 66, 4 (2018), 1594–1608.

[7] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. 2013. Mobile cloud computing: A survey. *Future generation computer systems* 29, 1 (2013), 84–106.

[8] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. 2016. Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds. *IEEE Transactions on Cloud Computing* (2016).

[9] Erol Gelenbe, Ricardo Lent, and Markos Douratsos. 2012. Choosing a local or remote cloud. In *2012 Second Symposium on Network Cloud Computing and Applications*. IEEE, 25–30.

[10] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center.. In *NSDI*. 22–22.

[11] Mike Jia, Jiannong Cao, and Weifa Liang. 2015. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing* 5, 4 (2015), 725–737.

[12] Mike Jia, Weifa Liang, Zichuan Xu, and Meitian Huang. 2016. Cloudlet load balancing in wireless metropolitan area networks. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[13] Yanchen Liu, Myung J Lee, and Yanyan Zheng. 2015. Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. *IEEE Transactions on Mobile Computing* 15, 10 (2015), 2398–2410.

[14] Pavel Mach and Zdenek Becvar. 2017. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1628–1656.

[15] Deepak Puthal, Mohammad S Obaidat, Priyadarsi Nanda, Mukesh Prasad, Saraju P Mohanty, and Albert Y Zomaya. 2018. Secure and sustainable load balancing of edge data centers in fog computing. *IEEE Communications Magazine* 56, 5 (2018), 60–65.

[16] Deepsubhra Guha Roy, Debashis De, Anwesha Mukherjee, and Rajkumar Buyya. 2017. Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. *The Journal of Supercomputing* 73, 4 (2017), 1672–1690.

[17] Mahadev Satyanarayanan, Victor Bahl, Ramón Caceres, and Nigel Davies. 2009. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing* (2009).

[18] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.