Evaluating Computation and Data Placements in Edge Infrastructures through a Common Simulator

Anderson Andrei da Silva, Clément Mommessin, Pierre Neyron, Denis Trystram, Adwait Bauskar, Adrien Lebre, Alexandre Van Kempen, Yanik Ngoko, Yoann Ricordel

Evaluating Computation and Data Placements in Edge Infrastructures through a Common Simulator

Anderson Andrei Da Silva
Institute of Mathematics and Statistics, University of São Paulo, Brazil
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG
Grenoble, France
anderson.andrei.silva@usp.br

Clément Mommessin, Pierre Neyron, Denis Trystram
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG
Grenoble, France
{firstname.lastname}@inria.fr

Adwait Bauskar, Adrien Lebre, Alexandre Van Kempen
Inria, LS2N, UMR 6004, IMT Atlantique
Nantes, France
{firstname.lastname}@inria.fr

Yanik Ngoko, Yoann Ricordel
Qarnot Computing
Montrouge, France
{firstname.lastname}@qarnot-computing.com

***Abstract*—Scheduling computational jobs with data-sets dependencies is an important challenge of edge computing infrastructures. Although several strategies have been proposed, they have been evaluated through ad-hoc simulator extensions that are, when available, usually not maintained. This is a critical problem because it prevents researchers to –easily– perform fair comparisons between different proposals.**

In this paper, we propose to address this limitation by presenting a simulation engine dedicated to the evaluation and comparison of scheduling and data movement policies for edge computing use-cases. Built upon the Batsim/SimGrid toolkit, our tool includes an injector that allows the simulator to replay a series of events captured in real infrastructures. It also includes a controller that supervises storage entities and data transfers during the simulation, and a plug-in system that allows researchers to add new models to cope with the diversity of edge computing devices.

We demonstrate the relevance of such a simulation toolkit by studying two scheduling strategies with four data movement policies on top of a simulated version of the Qarnot Computing platform, a production edge infrastructure based on smart heaters. We chose this use-case as it illustrates the heterogeneity as well as the uncertainties of edge infrastructures.

Our ultimate goal is to gather industry and academics around a common simulator so that efforts made by one group can be factorised by others.

***Index Terms*—Edge, Simulation, Scheduling algorithms, Data movements**

I. Introduction

The proliferation of Internet of Things (IoT) applications [1], as well as the advent of new technologies such as Mobile Edge computing [2], and Network Function Virtualisation [3] (NFV) have been accelerating the deployment of Cloud Computing-like capabilities at the edge of the Internet. Referred to as the Edge Computing [4] paradigm, the main objective is to perform on demand computations close to the place where the data are produced and analysed to mitigate data exchanges and to avoid too high latency penalties [5]. Among the open questions our community should address to favour the adoption of such infrastructures is the computation/data placement problem, i.e., *where to transfer data-sets according to their sources and schedule computations to satisfy specific criteria*. Although several works have been dealing with this question [6]–[12], it is difficult to understand how each proposal behaves in a different context and with respect to different objectives (scalability, reactivity, etc.). In addition to having been designed for specific use-cases, available solutions have been evaluated either using *ad hoc* simulators or through limited in-vivo (i.e., real-world) experiments. These methods are not accurate and not representative enough to, first, ensure their correctness on real platforms and, second, perform fair comparisons between them.

Similarly to what has been proposed for the Cloud Computing paradigm [13], we claim that a dedicated simulator toolkit to help researchers investigate Edge scheduling strategies should be released soon. Indeed, we claim that using placement simulators for Cloud Computing is not appropriate to study Edge challenges. Besides resource heterogeneity, network specifics (latency, throughput), and workloads, Edge Computing infrastructures differ from Cloud Computing platforms because of the uncertainties: connectivity between resources is intermittent, storage/computation resources are more heterogeneous and can join or leave the infrastructure at any time, for an unpredictable duration. In other words, a part of the infrastructure can be isolated or unavailable for minutes/hours preventing accessing some data-sets or assigning new computations.

In this article, we present several extensions we implemented on top of the Batsim/SimGrid toolkit [14], [15] to favour fair evaluations and comparisons between various scheduling and data placement strategies for Edge infrastructures. In particular, we developed an external module to allow injecting in the simulation any type of unforeseen events that could occur (e.g., a machine became unavailable at time t). We also implemented a Storage Controller to supervise all

transfers of data-sets within the simulated platform. We chose to rely on Batsim/SimGrid instead of any other available Edge simulators [16], [17] for the following reasons:

- Batsim has been especially designed to test and compare resource management policies in distributed infrastructures. In other words, the design of Batsim enforces researchers to use the same abstractions and, thus, favours straightforward comparisons of different strategies, even if they have been implemented by different research groups;
- Batsim promotes separation of concerns and enables the decoupling between the core simulator and the scheduler. Moreover, Batsim provides APIs in different languages (including Python, C++ and Rust) that makes the development of a scheduling strategy accessible for a large number of researchers;
- The accuracy of the internal models (computation and network) of SimGrid has been already validated [18], [19] and extensively used [20];
- SimGrid provides a plug-in mechanism, which is of particular interest to deal with the diversity of Edge devices: it lets researchers add new models of specific Edge facilities without requiring intrusive modifications into the simulation engine.

By extending Batsim to the Edge paradigm, we target a tool that will enable researchers/engineers to re-evaluate major state-of-the-art load balancing strategies. In particular, we think about scheduling strategies that have been proposed in desktop computing platforms, volunteer computing and computational grids [21], [22] as these infrastructures have several characteristics in common with Edge platforms.

To demonstrate the relevance of our proposal, we discuss several simulations we performed for the *Qarnot Computing* [23] use-case. The infrastructure of *Qarnot Computing* is composed of 3,000 disk-less machines (smart heaters) distributed across several sites in France. Each computing resource can be used remotely as traditional Cloud Computing capabilities or locally to satisfy data processing requirements of IoT devices deployed in the vicinity of the computing resource. As such, the *Qarnot* platform is a good example of Edge infrastructure, with computing units and mixed local/global jobs with data-sets dependencies.

The strategies presented in this article are simple. They aim to illustrate what can be done without important efforts. More advanced strategies can be analysed in the same manner. We are, for instance, investigating more advanced strategies that consider pulling data-sets from other Edge resources rather than from the centralised storage system of the *Qarnot Computing* infrastructure.

The rest of the paper is structured as follows. Section II gives an overview of the Batsim/SimGrid toolkit and the extensions we implemented. Section III presents the *Qarnot Computing* use-case and describes how we simulated this case study. Section IV discusses a first analysis of different scheduling strategies for the *Qarnot* platform. Section V presents the related work. Conclusion and future works are given in Section VI.

II. A Dedicated Simulator for Edge Platforms

Our proposal relies on extensions developed in the Batsim/SimGrid toolkit [15]. Released in 2016, Batsim delivers a high-level API on top of SimGrid [14] to ease the development and simulation of resource management algorithms. Thus, our proposal relies on tools already validated by our community.

A. Operational Components

1) SimGrid: SimGrid [14] is a state-of-the-art simulation toolkit that enables the simulation of distributed systems. SimGrid's relevance in terms of performance and validity has been backed-up by many publications [20]. In addition to providing the program to be evaluated, performing simulations with SimGrid requires writing a platform specification and interfacing the program to simulate. SimGrid enables the description of complex platforms, such as hierarchical infrastructures composed of many interconnected devices with possibly highly heterogeneous profiles, such as the edge ones.

2) Batsim and the decision process: Batsim [15] is a simulator engine built on top of SimGrid. It proposes a specialised API to help researchers design and analyse jobs and I/O scheduling systems. Such systems are for instance Batch Schedulers *a.k.a.,* Resource and Jobs Management Systems, in charge of managing resources in large-scale computing centres. Batsim allows researchers to simulate the behaviour of a computational platform in which workloads are executed according to the rules of a decision process. It uses a simple event-based communication interface: as soon as an event occurs, Batsim stops the simulation and reports what happened to the *decision process*.

The decision process embeds the actual scheduling code to be evaluated. In other words, to simulate a given scheduling algorithm, an experimenter has to implement this decision process. Comparing different algorithms consists in switching between different decision processes, which is easy in Batsim. Internally, the decision process (i) reacts to the simulation events received from Batsim, (ii) takes decisions according to the given scheduling algorithm, and (iii) drives the simulated platform by sending back its decisions to Batsim. Batsim and the decision process communicate via a language-agnostic synchronous protocol. In this work, we used Batsim's Python API to implement the decision process. For more details on Batsim and SimGrid mechanisms, we invite the reader to refer to Chapter 4 of Poquet's manuscript [24].

B. Extensions

To ease the study of scheduling and data placement strategies for Edge platforms, we have been working on a couple of extensions for Batsim. We present in this section those already available, namely the external events injector and the storage controller. Modifications made in Batsim and its Python API for this work are integrated in the main branch of

the repositories[1]. Besides, we present the plug-in mechanism of SimGrid that researchers can leverage to provide models of particular Edge devices.

1) External events injector: To simulate the execution of an Edge infrastructure, which is by essence subject to very frequent unexpected or unpredictable changes, our simulator offers the opportunity to inject external events on demand. Those events impact the behaviour of the platform during the execution and thus the choices of the scheduling strategy. For example, one would be interested in studying the behaviour and resilience of a scheduling policy when a range of machines becomes unexpectedly unavailable for a period of time, due to a failure or action (e.g., from a local user) occurring at the edge.

An external event is represented as a JSON object composed of two mandatory fields: a $timestamp$ that indicates when the event occurs, and the $type$ of the event. Depending on the type of event, other fields can complement the event description, such as for instance the name of the unavailable resource, the new value of an environment parameter such as the network bandwidth, or anything of interest to the decision process.

Similarly to the workload submissions, external events are replayed thanks to the injector process of Batsim. For each external event file given as input to Batsim, with one aforementioned JSON object per line in the file, an *external events submitter* is created during the initialisation of Batsim. Each submitter parses the list of external events from the input file and iterates over the list to submit the external events to the main process of Batsim at the right simulation times. Then, the external event is processed by Batsim, the state of the platform is updated and the occurring external event is forwarded to the decision process.

This event injection mechanism is generic by concept: users can define their own types of event and associated fields, which will simply be forwarded to the decision process without requiring any modification in the code of Batsim.

2) Storage Controller: The Storage Controller is a module included in Batsim's Python API to ease the management of storage entities and data-sets, and supervise data transfers during the simulation.

At the beginning of the simulation, the Storage Controller retrieves the list of *storage resources* of the platform and initialises one storage object per resource. These created storages are empty by default, but they can be filled on demand by the decision process by providing a single or a list of data-sets to be added to a storage. A data-set is represented by two fields, id and $size$, denoting the unique identifier of the data-set and its size in bytes. The Storage Controller exposes to the decision process an API to add data-sets to storages during the initialisation of the simulation. It also exposes functions to ask, for example, for the copy of a data-set from one storage to another, or to retrieve the list of all storages holding a copy of a given data-set during the simulation.

[1]https://gitlab.inria.fr/batsim/batsim and https://gitlab.inria.fr/batsim/pybatsim

When a data-set should be copied from one storage to another, the Storage Controller creates a specific Batsim job for data transfers describing that a given amount of bytes should be transferred from the source to the destination storage resource. Once Batsim notifies that this job completed, the Storage Controller notifies back the decision process that the requested data transfer has completed.

A timestamp is saved for each data movement. In other words, there is a timestamp associated to each data-set in each storage. This timestamp corresponds to the last time the data-set has been requested on this storage.

When adding a new data-set to a storage, the Storage Controller makes sure that there is enough available space in the destination storage before starting the data transfer. In the case there is not enough space, an eviction policy is used to determine which data-sets should be removed to free space for the new data-set. The default policy in use is LRU (*Least Recently Used*), which removes the data-set with the smallest timestamp in the storage. However, this eviction policy can be easily overridden by end-users of our simulator without diving into the main code. When implementing their decision process, end-users should simply create a call that inherits from the Storage Controller and override the eviction method. This enables the evaluation of more advanced eviction policies that can impact the overall scheduling decisions.

Finally, the presence of a particular data-set on a storage can be enforced through the Storage Controller API by assigning meta-information on a data-set. This information can then be used by the eviction policy to prevent for instance the deletion of the data-set while being used by running jobs.

3) SimGrid plug-ins: When designing an Edge simulator, it is a nonsense to foresee all the models and devices that may compose the platform. There are just too many. However, we claim that leveraging generic models is not the right solution either and so a trade-off should be found. We propose to leverage the SimGrid plug-ins capability that facilitates the implementation of new models without requiring intrusive changes in the simulation engine. We underline that, unfortunately, there is no generic manner of exposing information that can be captured by new plug-ins to the scheduler. Hence, some modifications might be required to extend the communication protocol of Batsim and exchange information between a particular plug-in and the decision process. Examples of such modifications are explained for the case of the *Qarnot* platform in Section III-D. This is the trade-off to be able to cope with the high heterogeneity of edge infrastructures while targeting accuracy of sub-models.

III. Case Study: the Qarnot Computing Platform

A. Infrastructure Overview

Qarnot Computing has been incorporated in 2010 to develop "a disruptive solution able to turn IT waste heat into a viable heating solution for buildings". The infrastructure is distributed in housing buildings, offices and warehouses across several geographical sites in France. As of writing this paper, the whole platform is composed of about 1,000 computing

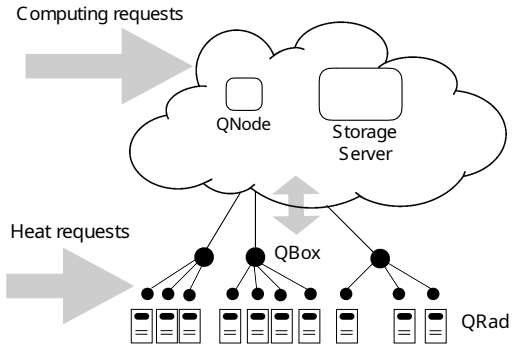

Fig. 1. Scheme of the Qarnot platform.

devices hosting about 3,000 disk-less machines, and is growing quickly. On each of the 20 geographical sites, there is a NFS-based storage with a few TB of capacity that enables disk-less machines to manipulate data. In a typical configuration a computing machine has a 1 Gbps uplink to a common switch, which then has up to 40 Gbps uplink to the NFS server. The latency between a computing machine and the NFS server is of the order of 1 ms. The various deployment sites are connected to the Internet using either a public or enterprise ISP, with characteristics varying from 100 Mbps to 1 Gbps symmetric bandwidth to the Internet, and about 10 ms latency to French data centres used by *Qarnot* to host control and monitoring services, the central storage system, and gateways to its distributed infrastructure.

On a daily basis, *Qarnot* computing solution processes from a few hundred to several thousands of batch jobs and thousands of cores are provisioned for dedicated corporate customers, and up to tens of GB of data are replicated from central storage to Edge Computing sites.

B. Platform Organisation and Terminology

The jobs and resources manager of the *Qarnot* platform is based on a hierarchy of 3 levels, as shown in Figure 1: the *Qnode*-, the *QBox*- and the *QRad*-level. The QNode is a root node, a "global" server that takes placement decisions for the whole platform. It can be viewed as a load balancer for the platform. On the second level are the QBoxes, "local" servers in smart buildings that take scheduling decisions locally on their own computing nodes. Each QBox is in charge of a set of computing nodes, the QRads, which are composed of one or several disk-less computing units denoted by *QMobos*.

Moreover, a centralised storage server is present at the QNode-level, while each QBox has its own local storage disk. From a physical point of view, the QNode and the storage server are in the Cloud while QBoxes are distributed over smart buildings of several cities, while QRads among a building are distributed in different rooms.

The *Qarnot* platform receives two types of user requests: requests for computing and requests for heating. The computing requests describe the workload to be executed on the platform. They are made by users that first upload input data needed to execute their jobs (named *QTasks*) to the centralised server and upload a Docker image either to the centralised server or the Docker Hub. Then, they submit the QTasks to the QNode. A QTask can be decomposed as a bag of several *instances* that share the same Docker image and data dependencies, but with different command arguments. This can be used for example to process each frame of a given movie, with one frame or a range of frames per instance.

The heating requests are made by inhabitants that can turn on and off the smart heaters in their homes, or set a target temperature for rooms to be reached as soon as possible. Since the computing units in a smart heater are unavailable when cooling is necessary, and are available otherwise, such changes increases the heterogeneity challenges of an Edge infrastructure: the computation capacity does not simply appear or disappear but also vary according to the heating needs.

C. Principle of the Actual Scheduling Policy

QTasks submitted to the platform are scheduled onto QMobos through two steps. The first step takes place at the QNode-level. The QNode greedily dispatches as much instances of the QTasks as possible on QBoxes, depending on the amount of QMobos available for computation on each QBox. The second step takes place at the QBox-level. Upon receiving instances of a QTask, the QBox will select and reserve a QMobo for each instance and fetch from the storage server each missing data dependency before starting the instances.

Notice that, at all times, a *Frequency Regulator* runs on each QRad to ensure that the ambient air is close to the target temperature set by the inhabitant, by regulating the frequencies of the QMobos and completely turning off a QRad if it is too warm. Moreover, whenever there is no computation performed on the QMobos while heating is required, "dummy" compute-intensive programs are executed to keep the QRad warm.

Modelling such an infrastructure to identify improvement opportunities and analyse new scheduling strategies is something possible with our Batsim extensions.

D. Instantiation with our Simulator

We now present how we modelled and instantiated the *Qarnot* platform with our simulation toolkit, also depicted in Figure 2. Due to space limitation we limit our description to major elements. Further information are available in our technical report [25].

Each QMobo is simulated as a single-core SimGrid *host* (representing a machine) as they are the only computing units of the platform. We keep the same hierarchical structure of the *Qarnot* platform: QMobos belonging to the same QRad are aggregated in the same SimGrid *zone* (representing a network). Similarly, all QRads of a same QBox are aggregated in the same zone, as well as all QBoxes of the QNode. The simulation of storage spaces is done by adding special hosts which carry the Batsim *storage* role. Thus, in each QBox zone, there is one additional storage host for the QBox disk. Similarly, there is one storage host in the QNode zone to represent the storage server. All these details are modelled by the XML platform file given to SimGrid.

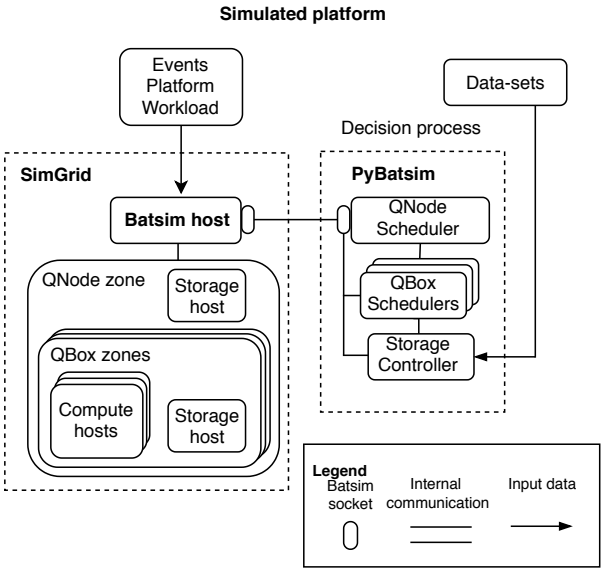

Fig. 2. Scheme of the simulated Qarnot platform.

Regarding the workload to simulate, each instance of a given QTask can run independently from the others, so we transcribed each instance as one Batsim job, with the same data-set dependencies and submission time for instances belonging to the same QTask.

As temperature plays an important role in the platform and the scheduling decisions, we leveraged the plug-in mechanism of SimGrid to implement our own model. Built on top of the existing energy plug-in [26], our plug-in computes the temperature of a QRad and its ambient air from the energy consumption of the QMobos and other physical parameters, such as the thermal conductivity and mass of the QRad. For the simulation of heating requests, each change of a QRad target temperature is simulated as an external event injected in the simulation. Besides, we take into account the outside temperature of the cities where the QRads are deployed. This value is measured on a one-hour basis. We modified Batsim to relay these external events to the plug-in, and we modified the communication protocol to periodically forward the ambient air temperature of each QRad to the scheduler. The modifications that are specific to the Qarnot use-case, such as the handling of temperature events, are available in a separate branch of the repository[2]. The temperature plug-in is also available in a separate SimGrid repository of the authors[3].

Finally, the schedulers of the QNode- and QBox-level were implemented using Batsim's Python API. The decision process is a Python process holding one instance of the QNode scheduler and the Storage Controller, and one instance of the QBox scheduler for each QBox of the simulated platform. Upon receiving a message from Batsim, the simulation events are directly forwarded to the correct destination scheduler or the Storage Controller that should handle it.

A log extractor was built to generate all the input JSON files from real logs of the *Qarnot* platform, for a given time period (including the list of jobs to execute, the list of data-sets in the storage server and the list of external events). It is noteworthy that due to users' privacy reasons, we cannot provide access to the log extractor and the *Qarnot* logs used for the experiments. Since we want to simulate an exact time period, we added a special external event that enforces the simulation to stop at a particular time.

IV. Simulations

Two kind of experiments have been performed to investigate the *Qarnot Computing* use case. The first aimed to compare the standard scheduling policy used in the real *Qarnot* platform with a policy based on locality of the data-sets. The second experiment enabled us to study the impact of replication policies for the data-sets that are uploaded on the platform (i.e., how they affect the scheduling decisions). The code of all evaluated schedulers is available in a dedicated branch of Batsim's Python API repository[4].

A. Data/Job Scheduling Policies

Along with the real *Standard Qarnot* scheduler that serves as a baseline for our experiments (see Section III-C), we implemented a variant using the data-locality to take scheduling decisions at the QNode-level, denoted by *LocalityBased*. Upon dispatching instances, LocalityBased gives priority to the QBoxes already having the data-set dependencies of the QTask on their storage disk. This variant aims at taking benefit from the data locality and reducing the data transfers.

To evaluate the impact of data placement on the scheduling decisions, we also implemented three variants of replication policies upon the submission of QTasks. The question we want to answer with these variants is whether replicating data-sets can achieve significant improvements, and at which cost? The first two variants, denoted by *Replicate3* and *Replicate10*, respectively replicates the data-dependencies of a submitted QTask on the 3 and 10 least loaded QBox disks among the 20 QBoxes in the platform, before applying the LocalityBased scheduling algorithm. These two variants aim at reducing the waiting time of the instances by providing more QBox candidates for the LocalityBased dispatcher. The last variant, denoted by *DataOnPlace*, instantaneously copies all data-set dependencies on all QBox disks upon the submission of a QTask. Even if it is unrealistic, this variant aims at visualising the behaviours of the standard scheduling policy without having any impact caused by the data transfers.

B. Simulated Workloads

We extracted 4 different simulation inputs corresponding to logs of the *Qarnot* platform for a 1-week period each. Since the simulation and the scheduling algorithms are deterministic, we ran one simulation with each combination of scheduler and

[2]https://gitlab.inria.fr/batsim/batsim/tree/temperature-sbac-2020

[3]https://github.com/Mommessc/simgrid/tree/temperature-sbac-2020

[4]https://gitlab.inria.fr/batsim/pybatsim/tree/temperature-sbac-2020

workload. Each simulation took less than 20 minutes to run, with about 60% of the time spent in the decision process.

The considered workloads contained between 5,000 and 9,000 instances and between 40 and 60 different data-sets. In each workload, there was at least one data-set used by 50% of the instances, and for the 2nd workload, up to 9 data-sets were used by 700 instances. This information shows that using replication for data-sets should improve the quality of the schedules compared to standard scheduling decisions.

In our simulations, we compared the quality of the produced schedules using the waiting time of the instances, the total number of transfers that occurred, and the total data transferred in GB. For one instance, the waiting time denotes the difference between its starting and submission times.

C. Simulation Results

Figure 3 and Figure 4 show respectively the waiting time distribution for each scheduling strategy and the amount of manipulated data we observed through simulations. Note that Figure 3 shows the waiting time distribution achieved by each scheduler for workload 3, separated in 3 intervals for better clarity. Due to lack of space and as other workloads showed similar results, we omit the corresponding figures and redirect the reader to our technical report [25].

1) Impact of data locality: As depicted on Figure 3, for each scheduler more than 60% of the instances waited less than one second before starting their execution. The last column shows that a few instances waited a long time before starting their execution (around 1,455 seconds). This is due to the long transfer time of one of their data dependencies that was as large as 36 GB, while other data-sets were smaller than 5 GB. Comparing the behaviour of Standard and LocalityBased, we do not observe a big difference in the distribution of the waiting times, except for the amount of instances that waited for 1,455 seconds. This is confirmed by the average value over all instances of 39 seconds for Standard and 34.6 for LocalityBased.

Regarding the amount of data manipulated Figure 4 shows, the results from the LocalityBased scheduler is as expected: dispatching instances on QBoxes already having the data-set dependencies on their disk permits to reduce the number of transfers by about 44%, and between 30 and 65% the total data transferred, compared to Standard.

To conclude, considering data locality decreases the amount of data transfer as expected but does not seem to be satisfactory enough to significantly improve waiting times.

2) Transferring data has a cost: Replicating data-sets permits to reduce the mean waiting times of the instances but at a cost of more data transfers, as depicted in Figure 4. More precisely for the 3rd workload, the mean waiting time of the instances decreases from 34.6 to 32.6, 28.6 and 22.2 seconds respectively for LocalityBased, Replicate3, Replicate10 and the unrealistic DataOnPlace strategies. While these results look encouraging, it is important to take into account the associated overhead in terms of data transfer: from 90 GB to 384 GB for Replicate3, 599 GB for Replicate10 and 1,056 GB for DataOnPlace. This respectively corresponds to an overhead in terms of data transfer of 4.3x, 6.7x and 11.7x.

Consequently, it is not clear whether replicating data-sets at a high ratio is a valid approach. On the first hand, taking into account only the data locality is not sufficient to have good waiting time performance (LocalityBased). On the second hand, it is crucial to control data-set exchanges as they have an impact on the overall performance. For instance, it may make sense to have a replication ratio that is dynamic according to the popularity of the data-set and the status of the platform. In other words, it is crucial to also consider the time spent in data transfer before taking scheduling decisions. This is critical as the size of data-sets should be increased with respect to IoT-based scenarios envisioned by *Qarnot Computing*. In this regard, we plan to extend the Storage Controller to estimate the transfer time of a data-set to a given storage entity at a certain time. This information is valuable for the schedulers to decide when triggering data transfers and on which QBoxes. Besides, we plan to leverage our proposal to evaluate whether exchanging data-sets directly between QBoxes can help us reduce the data transfer time.

Finally, we recall that our goal through this study was not to find the best scheduling algorithm but to illustrate the use of our simulation toolkit on a concrete scenario, and to demonstrate how such a simulator would help to drive the design of scheduling and data placement strategies. Capturing the aforementioned observations in the Qarnot Computing production platform would have been impossible.

V. Related Simulation Tools

We described in this paper a novel simulation tool for easily designing and testing scheduling and data placement strategies on Edge Computing platforms. We motivated the effort of building a new simulator using adequate tools for modelling the processing and storage units and the network topology. We discuss briefly below the main competitors and argument for our simulator.

Some simulators have constraints that would prevent us to correctly simulate a platform such as the *Qarnot* one. For example, EmuFog [27] does not support hierarchical fog infrastructures, whereas *Qarnot* infrastructure is inherently hierarchical.

Other simulators such as iFogSim [28], EdgeCloudSim [16] and IOTSim [17], are simulation frameworks that enable to simulate Fog or Edge Computing infrastructures and execute simulated applications on top of it. These solutions are close to our work. However they have been built on top of the CloudSim toolkit [29]. Although widely used to study algorithms and applications, CloudSim is based on a top-down approach of cloud environments. This is efficient to deliver the right abstractions to the end-users but unfortunately lacks of validations of the low-level models. We believe it is an important issue as it may return invalid observations. Besides, the Batsim/SimGrid toolkit is the only one that has been designed to study and compare scheduling challenges in an easy manner. In other simulators, including CloudSim, researchers have to

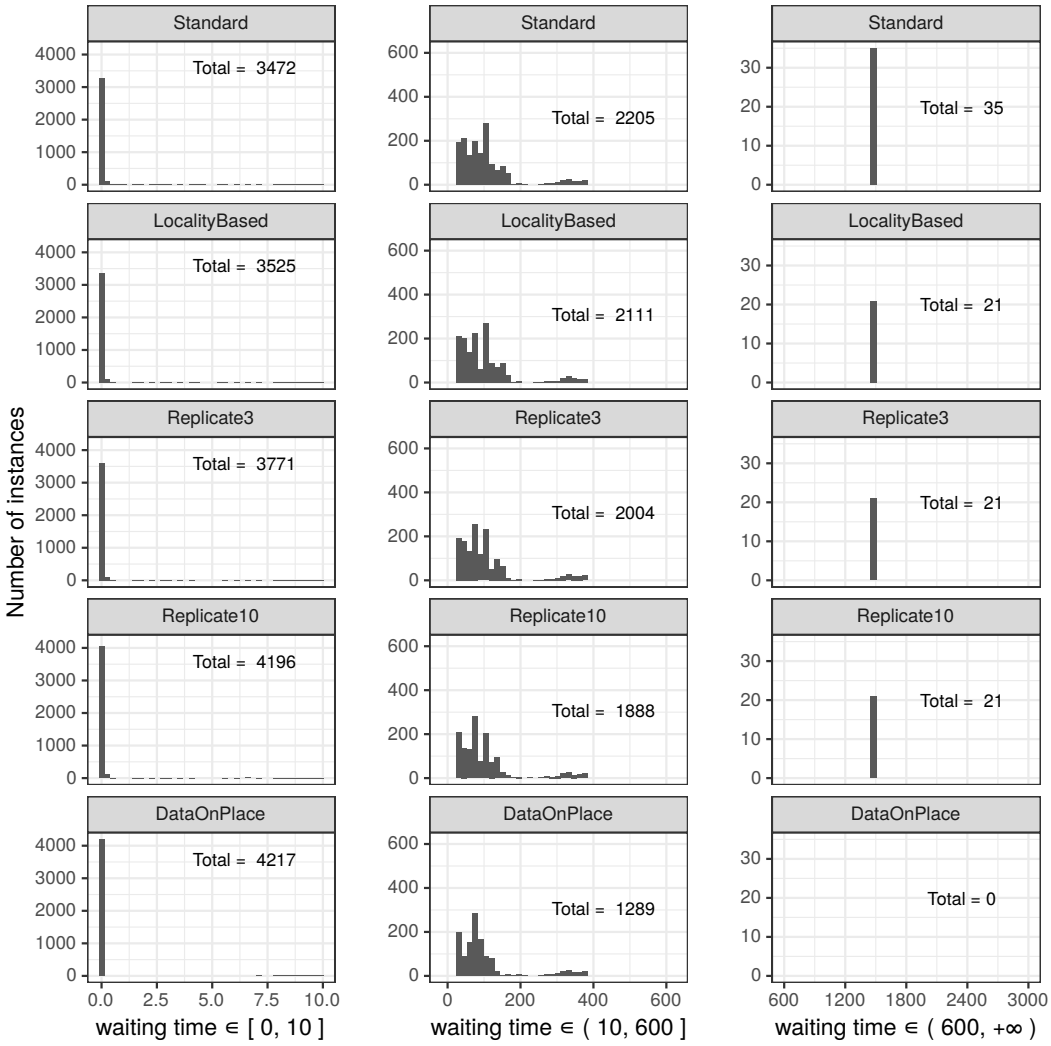

Fig. 3. Waiting time distribution (in seconds) of all instances of the 3rd workload.

implement a lot of business logic that is redundant each time they want to investigate a scheduling policy. Batsim/SimGrid delivers all this logic in a generic manner, making it more versatile and user-friendly for researchers/engineers.

VI. Concluding Remarks and Future Steps

We presented in this paper extensions we made to the Batsim/SimGrid framework to evaluate scheduling and data placement policies in Edge Computing infrastructures. Its integration into a simulator leads to a complete management system for Edge Computing platforms that focuses on the evaluation of scheduling strategies, taking into account both jobs and data.

While more extensions are still under development, the presented toolkit already enables researchers/engineers to easily evaluate existing load balancing and placement strategies. It may also serve at developing and testing new strategies thanks to its modular and clear interface.

To assess the interest of such simulator, we instantiated the toolkit to simulate the whole Edge platform of the *Qarnot Computing* company based on smart heaters. As a first use case, we investigated four scheduling strategies and compared them to the actual policy implemented in the *Qarnot* platform. We showed that replication of data-sets is an interesting approach to reduce job waiting times but requires additional investigations to determine how the replication ratio can be computed according to several metrics, such as data-set popularity, size, etc. To help researchers move forward on this question, we are currently extending the Storage Controller to monitor additional information such as the number of ongoing data transfers. We are also discussing with the SimGrid team to see how we can leverage information about the current load of the links between resources to have estimations of bandwidth, latency or the time to transfer a particular data-set from a source to a destination.

Besides, we envision to design an automatic and probabilistic injector of machine and network failures based on statistical

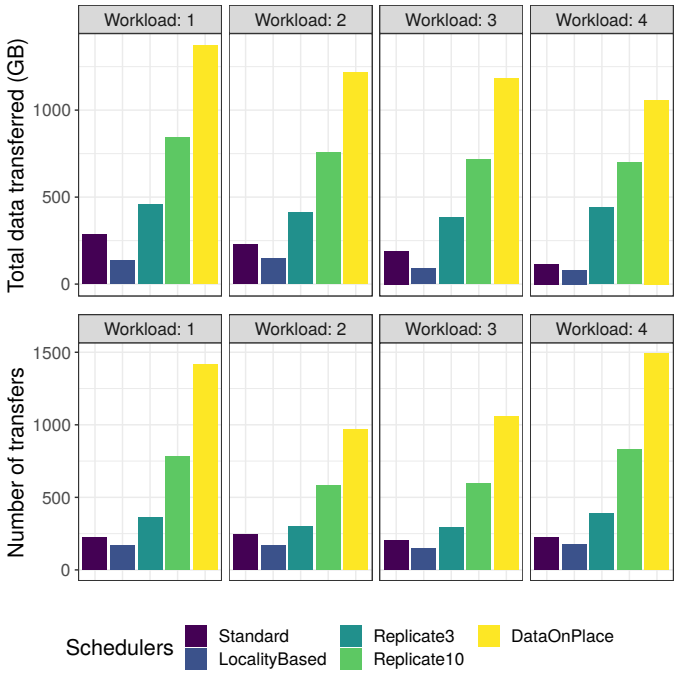

Fig. 4. Number of transfers and total data transferred in GB.

studies of the platform logs and learning techniques. Being able to model the dynamic of Edge infrastructures would be also an important added-value for our framework to capture side effects of such events on scheduling strategies.

ACKNOWLEDGMENTS

This work was supported by the ANR Greco project 16-CE25-0016-01 and by AUSPIN with the International Student Exchange Program from the University of São Paulo.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *2016 10th International Conference on Intelligent Systems and Control*, Jan 2016, pp. 1–8.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[5] B. Zhang, N. Mor, J. Kolb, D. Chan, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, and J. Kubiatowicz, "The Cloud is Not Enough: Saving IoT from the Cloud." in *HotStorage*, 2015.

[6] F. Ait Salaht, F. Desprez, A. Lebre, C. Prud'Homme, and M. Abderrahim, "Service Placement in Fog Computing Using Constraint Programming," in *SCC 2019 - IEEE International Conference on Services Computing*. IEEE, Jul. 2019, pp. 1–9.

[7] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, Oct 2017.

[8] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos, "Fog Based Framework for IoT Service Provisioning," in *IEEE CCNC*, Jan. 2019.

[9] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "iFogStor: An IoT Data Placement Strategy for Fog Infrastructure," in *ICFEC'17*, 2017, pp. 97–104.

[10] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT Service Placement in the Fog," *SOC*, vol. 11, no. 4, pp. 427–443, Dec 2017.

[11] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, and F. Desprez, "Combining Hardware Nodes and Software Components Ordering-based Heuristics for Optimizing the Placement of Distributed IoT Applications in the Fog," in *Proc. of the ACM SAC*, 2018, pp. 751–760.

[12] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "Qos-aware dynamic fog service provisioning," *CoRR*, vol. abs/1802.00800, 2018.

[13] A. Lebre, J. Pastor, A. Simonet, and M. Südholt, "Putting the next 500 vm placement algorithms to the acid test: The infrastructure provider viewpoint," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 204–217, Jan 2019.

[14] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, Jun. 2014.

[15] P.-F. Dutot, M. Mercier, M. Poquet, and O. Richard, "Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator," in *20th Workshop on Job Scheduling Strategies for Parallel Processing*, May 2016.

[16] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 39–44.

[17] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, "IOTSim: a Cloud based Simulator for Analysing IoT Applications," *J. Syst. Archit.*, vol. 72, no. C, pp. 93–107, Jan. 2017.

[18] A. Degomme, A. Legrand, G. Markomanolis, M. Quinson, M. L. Stillwell, and F. Suter, "Simulating MPI applications: the SMPI approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, p. 14, Aug. 2017.

[19] P. Velho, L. Schnorr, H. Casanova, and A. Legrand, "On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations," *ACM Transactions on Modeling and Computer Simulation*, vol. 23, no. 4, Oct. 2013.

[20] "SimGrid publications. http://simgrid.gforge.inria.fr/Publications.html."

[21] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data management and transfer in high-performance computational grid environments," *Parallel Computing*, vol. 28, no. 5, pp. 749–771, 2002.

[22] D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*. IEEE Computer Society, 2004, pp. 4–10.

[23] "Qarnot computing. https://www.qarnot.com."

[24] M. Poquet, "Simulation approach for resource management. (approche par la simulation pour la gestion de ressources)," Ph.D. dissertation, Grenoble Alpes University, France, 2017. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01757245

[25] A. Bauskar, A. da Silva, A. Lebre, C. Mommessin, P. Neyron, Y. Ngoko, Y. Ricordel, D. Trystram, and A. Van Kempen, "Investigating Placement Challenges in Edge Infrastructures through a Common Simulator," Feb. 2020, technical report. [Online]. Available: https://hal.inria.fr/hal-02153203

[26] F. C. Heinrich, T. Cornebize, A. Degomme, A. Legrand, A. Carpen-Amarie, S. Hunold, A.-C. Orgerie, and M. Quinson, "Predicting the Energy Consumption of MPI Applications at Scale Using a Single Node," in *Cluster 2017*. IEEE, Sep. 2017.

[27] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran, "Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures," in *FWC*. IEEE, 2017, pp. 1–6.

[28] H. Gupta, A. Vahid Dastjerdi, S. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments," *Software: Practice and Experience*, 06 2016.

[29] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software Practice and Experience*, vol. 41, pp. 23–50, 01 2011.