A Simulation-based Performance Evaluation of Heuristics for Dew Computing

Matías Hirsch, Cristian Mateos and Alejandro Zunino
ISISTAN – UNICEN – CONICET,
Tandil, Buenos Aires, Argentina
{matias.hirsch,cristian.mateos,alejandro.zunino}@isistan.unicen.edu.ar

Tim A. Majchrzak
University of Agder,
Kristiansand, Norway
timam@uia.no

Tor-Morten Grønli
Kristiania University College, Mobile Technology Lab
Dep. Technology, Oslo, Norway
tor-morten.gronli@kristiania.no

Hermann Kaindl
TU Wien, Inst. of Computer Tech.,
Vienna, Austria
hermann.kaindl@tuwien.ac.at

Abstract

The evolution of smartphones allows the continuous exploitation of computing resources. This increasingly applies also to distributed environments as exemplified through utilization of network router loads in edge computing and fog computing. Combining cloud computing and mobile smart devices in dew computing contexts enables new techniques for resource utilization, data collection and data processing. However, new challenges regarding job scheduling arise. Smartphones may be used in ad-hoc networks in this context, but their heterogeneity and energy usage must be considered. We propose novel heuristics for performance measuring of distributed computing systems integrated with mobile devices and compare them with previous heuristics in a simulation environment. Our results show an overall improvement in job completion and load balancing metrics compared to previous approaches. They highlight the usefulness of pursuing this research stream for aiming at industrial implementation and evaluation.

Keywords: Dew Computing, Edge Computing, Mobile Devices, Job Scheduling, Scheduling Heuristics

1. Introduction

Smartphones have increasing capabilities of processing information, which typically are *underutilized* [1]. Cities (and citizens) could benefit from such ubiquity of underutilized resources if these were properly orchestrated. Any person carrying a smartphone could contribute with valuable resources to help cities grow in a more sustainable way, and to manage them sustainably. For instance, anyone may help improving urban road maintenance by collecting pavement data [2]. Participatory platforms have been proposed to enable people to voluntarily contribute data sensed with their personal mobile devices [3, 4].

With IoT sensors and surveillance cameras installed, cities generate vast amounts of data for different smart city applications [5, 6]. Processing locally-sensed data can be done in different but not necessarily mutually exclusive ways. For instance, using distant cloud resources, offloaded to proximate fog servers or with the help of devices with computing capabilities within data collection context, e.g., with smartphones. This latter architectural option has also been referred to as an attractive self-supported sensing and computing scheme [7]. Hybrid and volunteer-supported processing architectures were also proposed as a way to avoid overloading resource-constrained devices [8]. Being the adopted approach hybrid or self-supported, managing smartphones' limited energy and heterogeneous computing capabilities requires more research [9].

In this work, we shed light on how to perform such resource management for a variant of self-supported sensing and computing scheme where data collected in a local context is processed by a group of smartphones within the same context. We call that a "dew context" to continue with the cloud / fog metaphor used to describe a layered computing infrastructure. Our contributions to the state of the art are: 1) we propose novel and practical load balancing heuristics that improve on the performance of previously proposed solutions [10, 11], 2) we illustrate our proposal and conduct its evaluation with a concrete dew context scenario, and 3) we compare the performance of the newly and previously proposed heuristics based on a set of metrics that have not yet been combined in related work.

This paper is organized as follows. Section 2 discusses related work and Section 3 gives a motivating example. After that, Section 4 presents our novel scheduling heuristics. Section 5 describes the evaluation methodology and experimental design, while Section 6 discusses the results. Lessons learned and limitations are stated in Section 7. Section 8 gives our conclusions.

2. Related Work

The exploitation of computing resources provided by smart devices in dew computing contexts — i.e., where both data collection and processing happen at the edge of the network — introduces new challenges in terms of job scheduling algorithms [9]. Since smart devices rely on batteries as their main power source, one of these challenges is to manage the remaining energy of resource provider nodes in the network. Basically, proposals must consider the impact of a given schedule on a device's future battery level. Including this aspect involves targeting the maximization of completed jobs without exceeding the node's energy availability.

There are at least two approaches for pursuing this objective. One models job scheduling as an optimization problem. Several authors [12, 13, 14] suggested to include a device's remaining energy as a constraint in the problem formulation. Such works explore feasible solutions where energy employed in executing jobs must not exceed the available energy on the devices' batteries. To tailor input variables of algorithms following this approach, it is necessary to have accurate job energy consumption data, which is impractical. To obtain such data in the general case, resource demand quantification details are needed, which, in turn, vary according to device characteristics. Given the wide variability of device models in the market (cf., e.g., [15]), it is unrealistic to assume homogeneous device clusters. If not pre-computed, scheduling input should be obtained while converting a data stream into jobs to be processed.

The other approach does not require energy-related job details. It performs load balancing based solely on node characteristics. Hirsch et al. [16] combined the last reported battery level with a function including different performance scores that rates the capability of a device to successfully complete the last arrived job. Jobs are scheduled by following an *online* approach, i.e., upon each job arrival, the scheduling logic creates a ranking by evaluating the function for all candidates devices; the job is assigned to the best ranked one.

Resource heterogeneity imposes other challenges that scheduling algorithms in dew computing contexts must deal with. The co-existence of smart devices that belong to the same or different generation, equipped with hardware able to render dissimilar computing and data transfer throughput, should not be ignored when including them as first-class resource providers. In [17], resource heterogeneity awareness is incorporated through information on the number of cores, speed and CPU workload that is evaluated by the proposed heuristics when allocating computing-intensive tasks to mobile devices. In [16], heterogeneity is considered by differentiating the nodes' computing capabilities via their MFLOPS indicator, which is a common metric in Scientific Computing to rate processor speed when executing floating-point operations. All in all, the heuristics in [17, 16] recognize resource heterogeneity related to computing capability only. For stream processing applications, where data transfer under varying delay and energy consumption of wireless communication is present, new practical online heuristics are necessary to deal with both node computing and communication heterogeneity.

3. Motivating Example

Smart Cities integrate multiple sources of information and process massive volumes of data to achieve efficient solutions and monitor the state of a wide range of common issues, including maintenance of public spaces and infrastructure or the security of citizens, just to mention two of them. Ultimately, they contribute to societal security [18]. Participatory sensing platforms encourage citizens to contribute incidents data, such as geolocalized photos, videos and descriptions that have to be analyzed, filtered and prioritized in a way for proper treatment. This requires a data processing infrastructure and depends on the citizens' willingness to manually enter or record data.

A proactive way to gather relevant data could be installing a dedicated sensor and processing infrastructure. However, to reduce fixed costs and to avoid the congestion of communication networks with a high volume of raw data captured [19], a hybrid approach that exploits near-the-field, ubiquitous computing power of smart mobile devices is feasible. By analyzing a city's dynamics, it is not hard to identify places where citizens are regularly connected to the same local area network with their smartphones, e.g., small parks or public transport. Suppose that citizens in such a context agree to contribute processing power while they may not like to provide data sensed with their devices. However, these may be used to filter and identify relevant information from data streams captured by sensors cleverly positioned within the context, and connected to the same network as nearby mobile users.

Consider, for instance, passengers riding a bus, where smartphones receive data via its WiFi. These may be samples of environmental sounds or images captured with devices that have been specifically installed in the bus for, e.g., real-time sensing of noise pollution, detecting pavement potholes, counting trees, or whatever information may be useful for a smart city to forecast events, schedule repairs or public space maintenance duties. The smartphones could be used, on

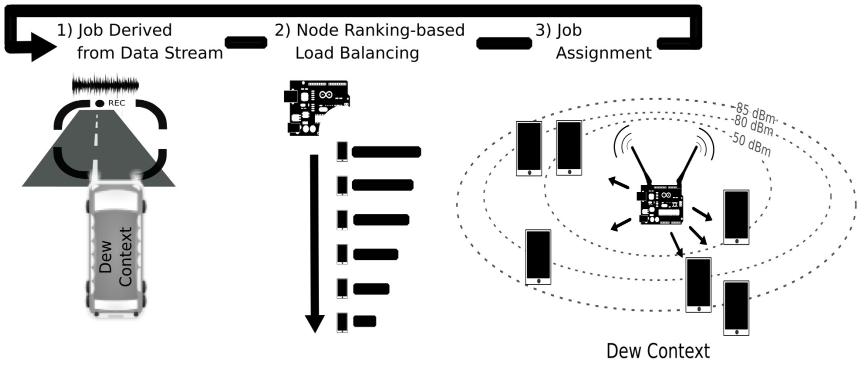

1) Job Derived from Data Stream
2) Node Ranking-based Load Balancing
3) Job Assignment
REC
Dew Context
-85 dBm
-80 dBm
-50 dBm
Dew Context

Figure 1. The dew context

a voluntary basis, for pre-processing such data before it is transferred to the distant Cloud in a curated form. How to efficiently and fairly balance data processing among available smartphones is a challenge, though.

4. Load Balancing Heuristics

Figure 1 depicts an overview of a *dew context*, a distributed computing mobile cluster (in this case operating inside a bus) for processing jobs locally generated. When close to the dew context local area network, mobile devices are enlisted to contribute with computing resources by registering themselves with the proxy [11]. In the example, the proxy is an on-chip-pc integrated circuit. The proxy balances jobs processing load with a heuristic that sorts devices' appropriateness using some criterion. The best ranked node is assigned with the incoming job and the ranking is re-built upon each job arrival. We propose and evaluate practical heuristics (criteria) to sort devices, which combine easy-to-obtain device and system performance information. One of these is AhESEAS, an improvement to the ESEAS (Enhanced Simple Energy-Aware Scheduler) [11]. Another criterion is ComTECAC, which was inspired by criteria targeting nodes' fair energy spending [20]. We now provide details of the formula components that these novel criteria combine to rank resource provider devices.

AhESEAS: the Ahead Enhanced Simple Energy Aware Scheduler is a criterion that combines a device's MFLOPS, its last reported battery level (SOC) and a counter of assigned jobs, in the same way as the ESEAS [11] formula, with the exception of a change in the semantics of the last mentioned counter. While in ESEAS the counter of assigned jobs is updated after the job input has been completely received by the node, in AhESEAS such an update occurs before, i.e., just after a node is selected for executing a job. By issuing an immediate counter update, i.e., without waiting for job input transferring time, gives AhESEAS rapid reaction to fast and continuous job generation typical for stream processing applications. To differentiate the semantic change and to avoid confusion with the ESEAS formula, we have renamed $assignedJobs$ of ESEAS by $queuedJobs$ in the AhESEAS formula (and adding 1 to avoid that the denominator may become 0):

$$AhESEAS = \frac{MFlops * SOC}{queuedJobs + 1}$$

ComTECAC: the Computation-Communication Throughput & Energy Contribution Aware Criterion utilizes indicators of a node's computing and communication capabilities, as well as its energy spent for executing dew jobs, which is implemented with battery level updates reported by nodes. Ranking heuristics using ComTECAC determines the best ranked node not only using a queued jobs component, but also with an energy contribution component. Thus, the load is evenly distributed among nodes, avoiding that strong nodes drain their batteries too much and earlier than weak nodes. The criterion's formula is:

$$ComTECAC =$$

$$\frac{MFlops * netPerf}{queuedJobs + 1} * (SOC - eContrib)$$

where:

- $MFlops$ and $queuedJobs$ have the same semantics as in AhESEAS.
- $netPerf$ is calculated as $\frac{1}{linkEfficiency}$, where $linkEfficiency$ relates node RSSI (received signal strength indicator) with Joules spent per KB of data transferred. $linkEfficiency$ is pre-computed and retrieved from a look-up table for a given RSSI value (see Table 2).
- SOC is a $(0, 1]$ normalized value of the last reported battery level.

- $eContrib$ is a $[0, 1]$ normalized value that accounts for the energy contributed by a device since it joined the mobile cluster. This value is updated upon each battery level drop reported by a node. The sub-formula to calculate it is $joinBattLevel - SOC$, where $joinBattLevel$ is the node battery level when it joined the cluster.

5. Evaluation: Methodology and Experiment Design

These new load balancing heuristics have been evaluated using DewSim [21], a simulation toolkit for studying scheduling algorithms' performance in clusters of mobile devices. Real data of mobile devices are exploited by DewSim via a trace-based approach that represents performance- and energy-related properties of mobile device clusters. This approach makes DewSim the best simulation tool so far to realistically simulate mobile device battery depletion, since existing alternatives use more simplistic approaches where battery depletion is modeled via linear functions. Moreover, the toolkit allows researchers to configure and compare scheduling performance under complex scenarios driven by nodes heterogeneity. In such a distributed computing system, cluster performance emerges, on one side, from nodes' aggregation operating as resource providers. On the other side, performance depends on how job scheduling logic manages such resources. A node's individual performance responds to node-level features including computing capability, battery capacity and throughput of the wireless link established with a centralized job handler (proxy). Table 1 and Table 2 outline node-level features considered in the experimental scenarios.

The computing-related node-level features presented in Table 1 refer to the performance parameters of real devices, whose brand/model is indicated in the first column. Such performance parameters include MFLOPS score that is used by the simulator to represent the speed at which jobs assigned by the scheduler are finalized. The MFLOPS of a device are calculated from averaging 20 runs of the multi-thread benchmark of the Linpack for Android app. The multi-thread version of the test causes a full occupation of all mobile device processor cores. The columns Node-type and OS version are informative, as these features are not directly configured in simulation scenarios but indirectly considered in the device profiling procedure. This procedure produces battery traces as a result, used to represent different devices' energy depletion curves.

Communication-related node-level features, i.e., time and energy consumed in data transferring events, such as job data input/output transferring and nodes status updates are shown in Table 2. Reference values correspond to a third-party study [22], which performed detailed measurements to characterize data transfer through WiFi interfaces, particularly the impact of received signal strength (RSSI) and data chunks size on time and energy consumption.

Device brand / model	Node-type	OS version	Mega FLOPS	Batt. max. capacity (joules)
LG / Optimus L9	smart-phone	Android 4.0	56	29 520
Samsung / Galaxy S3	smart-phone	Android 4.3	179	28 728
Acer / Iconia A100	tablet	Android 4.1.1	61	40 680
Phillips / TLE732	tablet	Android 7.1	104	33 300

Table 1. Computing-related node-level features

WiFi RSSI	dBm	send/receive			
		up to 10KB data		more than 10KB data	
		Through -put (KB/ms)	Energy (Joules / KB)	Through -put (KB/ms)	Energy (Joules / KB)
Excell.	-50	0.09	0.0099	0.4	0.00186
Good	-80	0.08	0.0106	0.25	0.00226
Mean	-85	0.04	0.0133	0.16	0.00333
Poor	-90	0.01	0.0346	0.025	0.01265

Table 2. Communication-related node-level features

Nodes ready to participate in a local, clustered computation form a mobile cluster at the edge whose computing capabilities derive from the number of aggregated nodes and their features. Cluster-level features considered in experimental scenarios are described in Tables 3 and 4. Specifically, Table 3 shows criteria to derive different types of heterogeneity levels w.r.t where the instantaneous computing throughput comes from. In short, targeting a defined QoS by relying on few nodes with high throughput differs in terms of potential points of failures and energy efficiency w.r.t to achieving this with many nodes having lower throughput each. Table 4 outlines criteria to describe communication-related properties of clusters where, for instance, an overall good communication quality –

Hetero-geneity	Rules
HtL0	100% of instantaneous computing capability provided by nodes of the same model
HtL1	74-76% of instantaneous computing capability provided by strong node models (relative to intracluster nodes)
HtL2	74-76% of instantaneous computing capability provided by weak node models (relative to intracluster nodes)

Table 3. Computing-related cluster-level features

GoodComm – means that a cluster has at least 80% of resource provider nodes with good or excellent RSSI[1]. In contrast, mean communication quality – MeanComm – suggests that a cluster has at least 60% of resource provider nodes with RSSI of -85 dBm. Finally, Table 5 shows the criteria used to conform cluster instances by combining the computation- and communication-related properties mentioned above. For instance, clusters of type Good2High are instances where nodes providing the fastest instantaneous computing capability relative to nodes in the cluster also have the best performance in terms of communication throughput. In contrast, the Good2Low category describes cluster instances where the best communication performance is associated with nodes able to provide the slowest instantaneous computing capability. Finally, the Balanced cluster category means that best communication performance is equally associated with nodes with the fastest and the slowest instantaneous computing capabilities.

Job sets were created using the *siminput* package utility of the DewSim toolkit [21]. We defined job bursts that arrive at varying intervals during a thirty minutes time window. Such a window represents a time extension where a vehicle can travel and scan a considerable part of its trajectory. Moreover, in this window the mobile devices of a group of passengers in a transport vehicle (e.g., a bus) can reasonably stay connected to the same shared access point. Intervals represent video or audio recording, i.e., in-bus data capturing periods. It is assumed that the recording system has a limited buffer, which is emptied at a point in time defined by a normal distribution with mean of 12 s and deviation of 500 ms. With every buffer emptying action, a new jobs burst is created and all captured data, which serves as input for a CPU-intensive program, is transferred to mobile devices that participate in the distributed processing of such data. Jobs are of a fixed input size. We created job sets where each job input has 1 MB and 500 KB, while output size varies between $[1 - 100]$ KB. Each job takes $[0.45 - 0.58]$ and $[1.43 - 1.85]$ s of computing time in the considered device model with the highest (Samsung Galaxy SIII) and lowest (LG L9) MFLOPS respectively. For defining time ranges, a pavement crack and pothole detection application implemented for devices with similar performance to those in the experiments of [2] was the reference. Bursts are composed of varying numbers of jobs, depending on the intervals extension.

[1]good_prop + mean_prop + poor_prop = 100% of nodes

Communnication Performance	Rules
Good Comm	good_prop: above 80% of nodes with Excellent/Good RSSI mean_prop: between 0% and (100% - good_prop) of nodes with Mean RSSI poor_prop: between 0% and 100% - (good_prop + mean_prop) of nodes with Poor RSSI
Mean Comm	good_prop: between 0% and 20% of nodes with Excellent/Good RSSI mean_prop: 100% - (good_prop + bad_prop) of nodes with Mean RSSI poor_prop: between 0% and 20% of nodes with Poor RSSI

Table 4. Communication cluster-level feature

@ Cluster type	Communication/computation nodes assignment criteria
@ Good2High	Good RSSI values are assigned firstly, among strong nodes. Remaining RSSI values among remaining nodes
@ Good2Low	Good RSSI values are assigned firstly, among weak nodes. Remaining RSSI values among remaining nodes
@ Balanced	25% of Mean RSSI values assigned to strong nodes, 25% of Mean RSSI values assigned to weak nodes and remaining RSSI values -good, mean, poor- are randomly assigned among remaining nodes

Table 5. Mapping of communication and computation cluster-level features

Figure 2 shows plots of volumes of data transferred, job counts, and MFLOPS required for processing such data and arrival time of each jobs burst for a period of 30 minutes. For example, jobs whose input data was set to 1 MB, represent a total of 52.78 GB of data requiring approximately 4 775 GFLOPS to be executed.

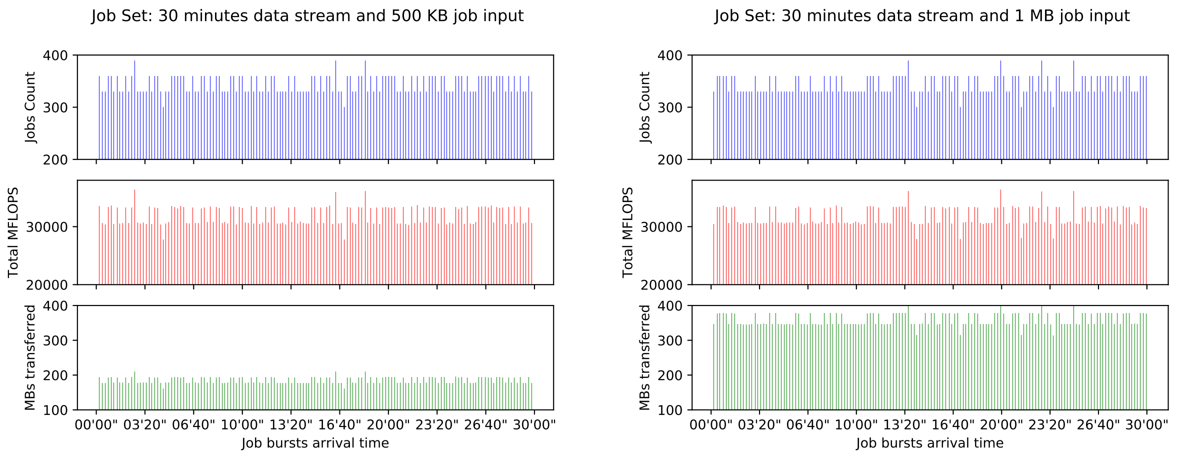


Figure 2. Job set characteristics

6. Metrics and Results

The scheduling heuristics' performance is measured in terms of completed jobs, makespan and fairness, which are metrics reported in similar distributed computing systems studies [17, 23, 16, 24].

Completed jobs: Providing that mobile device clusters rely on the energy stored in the mobile devices' batteries to execute jobs, scheduling technique A is considered to be more energy efficient than scheduling technique B, if the former completes more jobs than the latter with the same amount of energy. The jobs completion count finishes when all nodes leave the cluster, in this case, due to running out of energy.

Makespan: Measures the time the distributed system needs to complete the execution of a job set. We normalized these times duration into a $[0-1]$ scale, where the value 1 refers to the heuristic that requires the longest makespan. To calculate the makespan, we compute the difference between the time when the first job arrives and the time when the last job is completed. To calculate the latter when all compared heuristics achieved different numbers of completed jobs, we first compute the maximum amount of jobs that all heuristics completed, and use this value as a pivot to obtain the time when the last job is completed.

Fairness: The difference in energy contributed by provider nodes from the time each one joins the cluster to the time each one completes its last assigned job, is quantified via the Jain's Fairness index [25]. This index has been originally used to measure the bandwidth received by clients of a networking provider but, in our case, much as in [13, 24], it is used to measure disparity of energy pulled by the system from provider nodes. The metric complements the performance information given by completed jobs and makespan metrics.

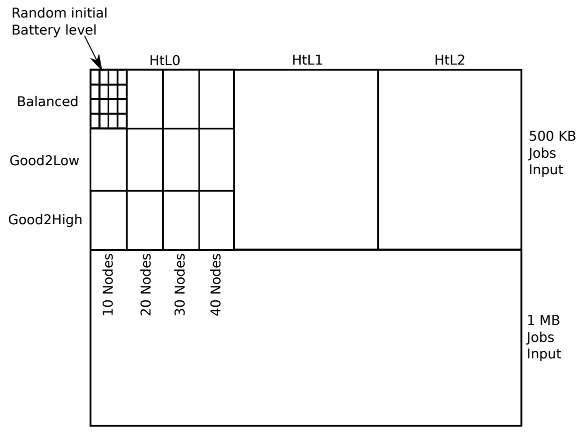
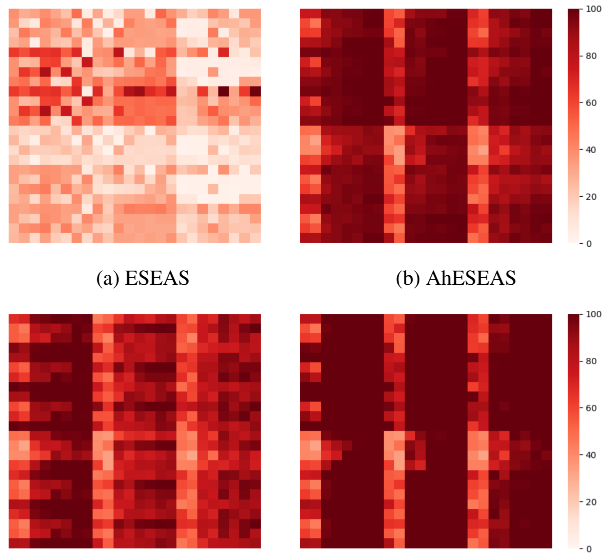


Figure 3. Simulated scenarios: Heatmap pixels

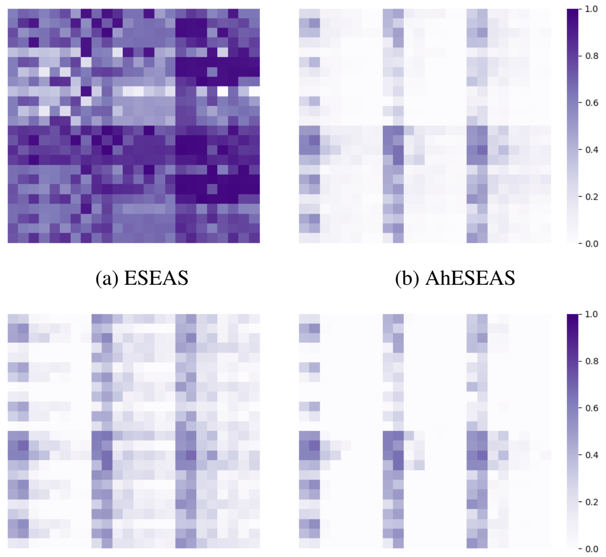
We tested all heuristics on 1 152 scenarios with varying nodes, cluster and jobs characteristics. Figure 3 depicts the position that each group of scenarios occupies in the heatmap representation used to display the performance values obtained for each heuristic.

Figure 4 shows the results of each heuristic's completed jobs for all scenarios described above. The darker the pixel intensity, the better is the performance achieved. Several effects of simulated variables on completed jobs are observed. First, by comparing Figure 4b and Figure 4a, which show the numbers of completed jobs for AhESEAS and ESEAS, respectively, we see the magnitude of improvement introduced by the AhESEAS denominator component update policy (see Section 4). In the presence of job input above 500 KB and approximately 360 jobs generated every 12 seconds, which is the injected load in the scenarios, load balancing is better managed by the denominator update policy of the AhESEAS formula than the one of ESEAS. AhESEAS exceeds the numbers of completed jobs of ESEAS in all scenarios. On average, AhESEAS was



(a) ESEAS (b) AhESEAS (c) RTC (d) ComTECAC

Figure 4. Completed Jobs (dark is better)



(a) ESEAS (b) AhESEAS (c) RTC (d) ComTECAC

Figure 5. Queued Jobs at proxy-side (light is better)

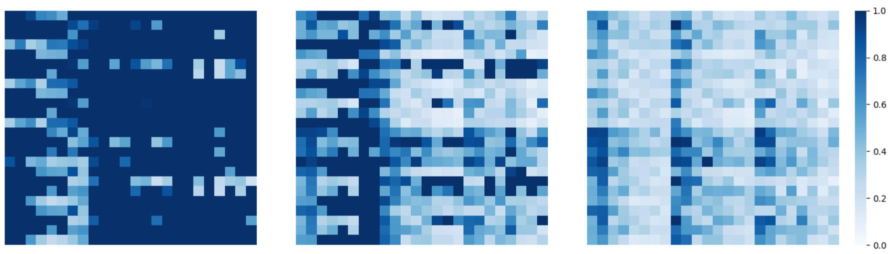
better by 58.6% with a standard deviation of 18.5%.

Due to the poor performance of ESEAS, we focus below on presenting results of AhESEAS, RTC and ComTECAC. Orthogonal to all heuristics, we see how completed jobs decreases as job input size increases. This is more noticeable in load balancing performed with AhESEAS than with RTC and ComTECAC. When comparing the scenarios in Figure 4b with job inputs of 1 Mb (bottom half) and job inputs of 500 Kb (top half), we see that the scenarios in the first group are lighter than in the second, and this can be attributed to the fact that the RTC and ComTECAC ranking formulas include communication-related parameters. RTC uses job data input/output size and nodes RSSI, while ComTECAC utilizes a function of nodes RSSI that relates this parameter with network efficiency. Please refer to [10] for more details of the RTC ranking formula.

Other variables with orthogonal effect in the amount of completed jobs are cluster size and cluster heterogeneity. The cluster size effect is notorious in Figures 4b, 4c and 4d, where 10 nodes cluster size scenarios completed fewer jobs than scenarios with 20, 30 and 40 nodes. Moreover, cluster heterogeneity degrades the performance of AhESEAS and RTC more than the one of the ComTECAC heuristic. In other words, AhESEAS and RTC do not manage as well as ComTECAC, the presence of weak nodes contributing with most of the instantaneous cluster computing capability. Finally, by taking completed jobs as an indicator of energy-harvesting capability, it can be seen that ComTECAC is the most energy-efficient scheduling heuristic on average. It completes more jobs than all the other heuristics, given the same amount of energy.

As jobs are being generated, the scheduling logic assigns them to nodes to be executed. Under this scheme, a job passes through at least three phases until it is completed. We model an input transfer phase when job input data is transferred from the proxy to the device in charge of its execution, an executing phase when the code associated with the job is executed, and an output transfer phase when device transfers the result of a job to the proxy node. This helped us to uncover the cause that leads jobs to be non-completed. As Figure 5 shows, irrespective of the heuristic, there is a strong correlation between the non-completed jobs state and the input transfer phase, which, in turn, is associated with job input size. The bigger the job input, the higher is the number of non-completed jobs.

Figure 6 depicts AhESEAS, RTC and ComTECAC performance measured in terms of makespan for all scenarios. In this case, pixels color intensity inversely relates to high performance, i.e., the lighter the pixel the better the performance. As explained above, where the metrics are presented, to report a comparable measurement between all heuristics, and only for scenarios where all heuristics differ in the amount of completed jobs, we computed makespan, considering a subset of completed jobs, instead of all jobs, in the following manner. For each scenario, we figured out the maximum of completed jobs for all heuristics and used this value as a reference to compute the makespan value of each heuristic. ESEAS was left aside from the analysis due to its overall poor performance in terms



(a) RTC (b) AhESEAS (c) ComTECAC

Figure 6. Makespan (light is better)

of completed jobs. Then, makespan values reported in Figure 6 were calculated using completed jobs of the RTC, AhESEAS and ComTECAC heuristics. The dark blue pixels predominance in Figure 6a indicates that RTC was the heuristic with the overall longest makespan on average, compared to that obtained by the AhESEAS and ComTECAC heuristics. Effects of job input size, cluster size and cluster heterogeneity described for the completed jobs metric still apply. Again, the advantage of ComTECAC over the other heuristics in the majority of scenarios is remarkable.

Finally, we report the performance of the heuristics using the fairness metric. Provided that the heuristics targeted different numbers of completed jobs for the same scenario, to calculate fairness, we followed similar initial steps as with makespan. For each scenario, we first determined the maximum number of completed jobs by all heuristics (except ESEAS), and used it as reference value to compute the fairness score. Once obtained, for each heuristic, we searched for the associated time stamp when this number of completed jobs has been reached. The time stamp is another reference, in this case, to get the last battery level reported by each participating node. Then, with such data, and the initial battery level reported by each node, we computed an energy delta, i.e., the node energy contribution, which is interpreted as a sample in the fairness score calculation formula.

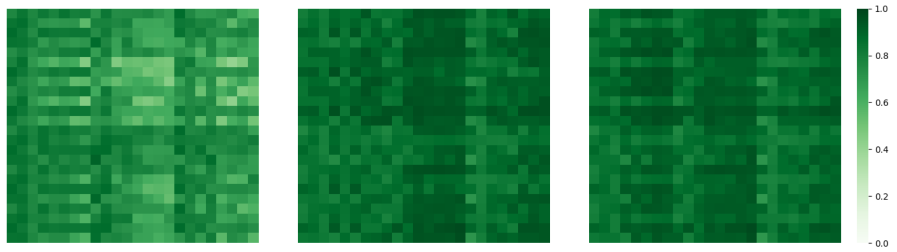
According to Figure 7, RTC achieves, on average, less fairness than AhESEAS and ComTECAC. However, we are not able to affirm the same when comparing AhESEAS and ComTECAC with each other. Hence, we formulated the null hypothesis H_0 that the fairness achieved by the last two heuristics is the same. We tested H_0 with the Wilcoxon test, pairing the fairness values of AhESEAS and ComTECAC for each scenario. This resulted in $p = 3.93 * 10^{-9}$, which lead us to reject H_0. To conclude this analysis and figure out which of the last two heuristics performed better, we re-computed the fairness metric, this time considering completed jobs only by the AhESEAS and ComTECAC heuristics. The results are shown in Figure 8a and Figure 8b, where ComTECAC achieves an apparently better performance than AhESEAS. We confirm that by complementing recomputed fairness heat maps with a cumulative scenarios density function of Figure 8c. Notice that the ComTECAC CDF increase is more pronounced than that of AhESEAS as the fairness score increases, i.e., for a good proportion of scenarios ComTECAC achieves higher fairness than AhESEAS.

7. Discussion

7.1. Lessons Learned

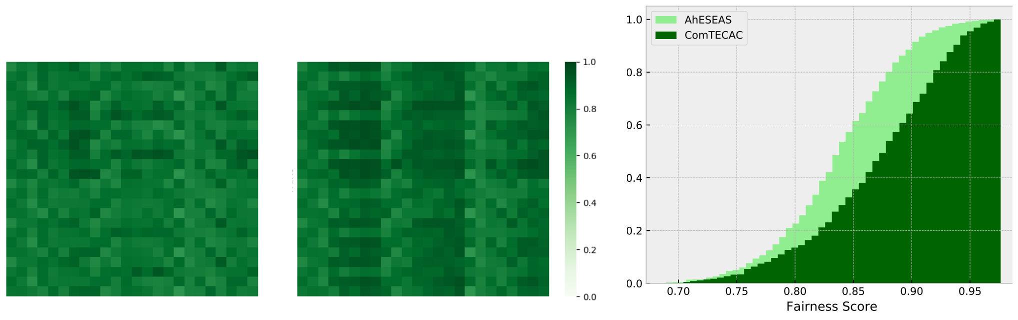
From the extensive experiments performed, and particularly by comparing the ESEAS and AhESEAS results, we can conclude that the change in the denominator update policy of AhESEAS, which updates the queued jobs component as soon as a node is selected for executing the next job, improves completed jobs for all simulated scenarios. This holds when job input size is at least 500 KB w.r.t. updating the denominator component when the node completed receives some job input. The improvement is 58.6%, on average.

Moreover, completed jobs of AhESEAS, which ranks nodes according to their computing capability, were on average 8.2% and 4.3% higher than for the RTC heuristic for 500 KB and 1 MB data input scenarios, respectively. RTC ranks nodes based on their transferring capability and queued data. For this reason, RTC performance seems to be less affected by jobs data transfer requirements and nodes transfer capabilities than AhESEAS and ComTECAC. For ranking nodes, the latter combines communication and computing capabilities, as well as energy contribution parameters. The combination of all these allows this heuristic to be better than all evaluated competitors in all metrics. It completes, on average, around 3% more jobs than the second best heuristic (AhESEAS) and achieves a speedup of 1.69 w.r.t. the second fastest heuristic (AhESEAS). ComTECAC is also the fairest heuristic for load balancing. For the first 50% (median), 80% and 90% distribution samples, the fairness value is 0.88, 0.92



(a) RTC (b) AhESEAS (c) ComTECAC

Figure 7. Fairness (dark is better)



(a) AhESEAS (b) ComTECAC (c) CDF

Figure 8. Recomputed fairness considering AhESEAS and ComTECAC heuristics only (dark is better)

and 0.94, respectively, while the second fairest heuristic (AhESEAS) for these cutting points results in fairness values 0.84, 0.86 and 0.87, respectively.

7.2. Limitations

Our collaborative computing scheme lacks mechanisms for promoting citizens' participation, accounting for computing contribution and preventing fraud in reporting results. Incentive mechanisms proposed for collaborative sensing are not applicable for resource-intensive tasks. Some of the questions that remain unanswered in this context are: Is the job completion event a good checkpoint for giving credits to resource provider nodes? What are the consequences of giving a fixed amount of credits upon a job completion irrespective of the time and energy employed by a device? How many results of the same job would be necessary to collect in order to prevent fraud in reporting job results?

Another limitation we are working on is a middleware prototype for validating our findings. We already integrated libraries that use traditional machine vision and deep learning object recognition and tracking algorithms into our device profiling platform, which is a satellite project of the DewSim toolkit. This is necessary to validate our load balancing heuristics with real object recognition algorithms, which in turn complements our battery-trace capturing method that currently exercises CPU floating-point capabilities through a generic yet synthetic algorithm. This integration also allows for deriving new heuristics to refine the exploitation of mobile devices by profiling specialized accelerator hardware such as GPUs and NPUs, which are suited for running complex AI models [26].

8. Conclusion

In this paper, we present a performance evaluation of novel practical job scheduling heuristics for stream processing in dew computing contexts. ESEAS and RTC are heuristics from previous work, while AhESEAS and ComTECAC are new. We measured the performance using the completed jobs metric, which quantifies how efficiently the available energy in the system is utilized; the makespan metric, which indicates how fast the system completes job arrivals; and the fairness metric, which measures the energy contribution differences among participating devices. The new heuristics had superior performance. These results present a step towards materializing the concept of dew computing using mobile devices from regular users. It will be applied to real-world situations where online data gathering and processing at the edge are important.

Despite our focus on heuristics to orchestrate a self-supported distributed computing architecture that leverages idle resources from clusters of battery-driven nodes, to extend the architecture' applicability, new efforts will follow. We will study complementing battery-driven resource provider nodes with non-battery driven fog nodes, e.g., single-board computers, in a similar way to other works studying the synergy among different distributed computing layers, e.g., fog nodes and cloud providers [27].

Acknowledgments

We acknowledge the financial support by CONICET (grant 11220170100490CO) and ANPCyT (grant PICT-2018-03323).

References

[1] D. Hintze, R. D. Findling, S. Scholz, and R. Mayrhofer, "Mobile device usage characteristics: The effect of context and form factor on locked and unlocked usage," in *Proc. 12th MoMM*, pp. 105–114, 2014.

[2] A. Tedeschi and F. Benedetto, "A real-time automatic pavement crack and pothole recognition system for mobile android-based devices," *Advanced Engineering Informatics*, vol. 32, pp. 11–25, 2017.

[3] F. Restuccia, S. K. Das, and J. Payton, "Incentive mechanisms for participatory sensing: Survey and research challenges," *ACM TOSN*, vol. 12, no. 2, pp. 1–40, 2016.

[4] H. Gao, C. H. Liu, W. Wang, J. Zhao, Z. Song, X. Su, J. Crowcroft, and K. K. Leung, "A survey of incentive mechanisms for participatory sensing," *IEEE Commun. Surv*, vol. 17, no. 2, pp. 918–943, 2015.

[5] V. Moustaka, A. Vakali, and L. G. Anthopoulos, "A systematic review for smart city data analytics," *ACM CSUR*, vol. 51, no. 5, pp. 1–41, 2018.

[6] C. Dobre and F. Xhafa, "Intelligent services for big data science," *Future generation computer systems*, vol. 37, pp. 267–281, 2014.

[7] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM CSUR*, vol. 48, no. 1, pp. 1–31, 2015.

[8] B. Ali, M. A. Pasha, S. ul Islam, H. Song, and R. Buyya, "A volunteer supported fog computing environment for delay-sensitive iot applications," *IEEE IoT Journ.*, 2020.

[9] R. Olaniyan, O. Fadahunsi, M. Maheswaran, and M. F. Zhani, "Opportunistic edge computing: Concepts, opportunities and research challenges," *FUTURE GENER COMP SY*, vol. 89, pp. 633–645, 2018.

[10] M. Hirsch, C. Mateos, J. M. Rodriguez, A. Zunino, Y. Garı, and D. A. Monge, "A performance comparison of data-aware heuristics for scheduling jobs in mobile grids," in *2017 XLIII CLEI*, pp. 1–8, IEEE, Sept 2017.

[11] M. Hirsch, J. M. Rodríguez, A. Zunino, and C. Mateos, "Battery-aware centralized schedulers for cpu-bound jobs in mobile grids," *Pervasive and Mobile Computing*, vol. 29, pp. 73–94, 2016.

[12] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive d2d collaboration for energy-efficient mobile edge computing," *IEEE Wireless communications*, vol. 24, no. 4, pp. 64–71, 2017.

[13] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," *Computers & Electrical Engineering*, vol. 44, pp. 218–240, 2015.

[14] X. Wei, J. Fan, Z. Lu, and K. Ding, "Application scheduling in mobile cloud computing with load balancing," *J Appl Math*, vol. 2013, 2013.

[15] C. Rieger and T. A. Majchrzak, "A Taxonomy for App-Enabled Devices: Mastering the Mobile Device Jungle," in *Revised Selected Papers WEBIST 2017*, vol. 322 of *LNBIP*, pp. 202–220, Springer, 2018.

[16] M. Hirsch, J. M. Rodríguez, C. Mateos, and A. Zunino, "A two-phase energy-aware scheduling approach for cpu-intensive jobs in mobile grids," *Journal of Grid Computing*, vol. 15, no. 1, pp. 55–80, 2017.

[17] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, and M. Imran, "Heterogeneity-aware task allocation in mobile ad hoc cloud," *IEEE Access*, vol. 5, pp. 1779–1795, 2017.

[18] P. Marana, C. Eden, H. Eriksson, C. Grimes, J. Hernantes, S. Howick, L. Labaka, V. Latinos, R. Lindner, T. A. Majchrzak, I. Pyrko, J. Radianti, A. Rankin, M. Sakurai, J. Sarriegi, and N. Serrano, "Towards a resilience management guideline – cities as a starting point for societal resilience," *Sustainable Cities and Society*, vol. 48, 2019.

[19] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *arXiv preprint arXiv:2005.10460*, 2020.

[20] J. Furthmüller and O. P. Waldhorst, "Energy-aware resource sharing with mobile devices," *Computer Networks*, vol. 56, no. 7, pp. 1920–1934, 2012.

[21] M. Hirsch, C. Mateos, J. M. Rodriguez, and Z. Alejandro, "Dewsim: A trace-driven toolkit for simulating mobile device clusters in dew computing environments," *Software: Practice and Experience*, 2019.

[22] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 29–40, 2013.

[23] P. Pandey and D. Pompili, "Exploiting the untapped potential of mobile distributed computing via approximation," *Pervasive and Mobile Computing*, vol. 38, pp. 381–395, 2017.

[24] H. Viswanathan, E. K. Lee, I. Rodero, and D. Pompili, "Uncertainty-aware autonomic resource provisioning for mobile cloud computing," *IEEE TPDS*, vol. 26, no. 8, pp. 2363–2372, 2014.

[25] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, DEC*, 1984.

[26] A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, and L. Van Gool, "Ai benchmark: All about deep learning on smartphones in 2019," in *2019 IEEE/CVF ICCVW*, pp. 3617–3635, IEEE, 2019.

[27] T. Elgamal, A. Sandur, P. Nguyen, K. Nahrstedt, and G. Agha, "Droplet: Distributed operator placement for iot applications spanning edge and cloud resources," in *IEEE 11th int. conf. CLOUD*, pp. 1–8, IEEE, 2018.