Optimizing Task Allocation for Edge Micro-Clusters in Smart Cities

Yousef Alhaizaey
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
y.alhaizaey.1@research.gla.ac.uk

Jeremy Singer
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
jeremy.singer@glasgow.ac.uk

Anna Lito Michala
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
annalito.michala@glasgow.ac.uk

***Abstract*—Current urban technology trends like Internet-of-Things and 5G require ultra low latency compute resource to be distributed liberally at the network edge. We characterize and advocate the need for heterogeneous edge micro-clusters; these are pragmatic, low-power, low-cost, minimal footprint units that can provide sufficient resource for typical edge compute applications in smart cities. However, to make best use of heterogeneous edge micro-clusters, we require resource management techniques that are both efficient and effective. In this paper, we report on an empirical study to demonstrate that mathematical optimization (in particular, mixed integer programming) for resource management is appropriate in terms of overhead, also highly effective for executing batch-arrival workloads in smart city use cases.**

***Index Terms*—Resource Management, Task Allocation, Internet of Things, Edge Computing, Edge Cluster**

I. Introduction

Edge compute is becoming mainstream, driven by growth in consumer adoption of 5G [1] and Internet-of-Things (IoT) technology [2] for smart city systems. Many end-user applications require low-latency data processing and interactive responses, motivating the provision of highly available edge compute capability. In this paper, we argue that the *heterogeneous edge micro-cluster* (HEμC) is a pragmatic instantiation of computing resource at the network edge. Further, we show that mathematical optimization techniques like integer programming are most appropriate for edge resource allocation scenarios in contrast to the high performance computing domain, given the relatively small scale of the solution space.

What is an Edge Micro-Cluster? Why is it heterogeneous? In a smart city, a HEμC might be a road-side unit in an autonomous vehicular system, a collection of smart home devices on a shared local area network, or an ad-hoc cluster of portable devices owned by a group of co-located individuals. Heterogeneity emerges naturally through partial infrastructure upgrades over time, installed device diversity and opportunistic clustering techniques. In our experiments, we model a typical HEμC using a single-board computer cluster, such as a stack of Raspberry Pi nodes.

Because of differences in infrastructure, architecture, resource distribution, connectivity and capacity at the edge, classical resource management methods must be reexamined. It is not evident that datacenter management techniques are directly transferable to edge scenarios. Section II-A explores the concept of HEμCs in greater detail, along with deployment motivation.

Why are Edge Micro-Clusters deployed in Smart Cities? There is a wide range of smart city use cases for edge computing. Generally, such applications are presented as high-level abstract scenarios such as self-driving cars, augmented reality tourism, or smart healthcare [3]–[5]. With respect to understanding concrete compute resource requirements, it is more practical to consider specific usage scenarios at a task processing level.

The DeFog [6] and Edge Bench [7] suites consist of idealized example tasks relating to smart city applications that may be executed by edge devices. In this paper, we argue that a HEμC is likely to receive a set of related compute requests as a batch of jobs, with a low-latency requirement for completion of all jobs in the batch. This might be object detection in a set of still photo images, for instance. Section II-B explores the batch job concept in greater detail, and outlines how edge benchmark suites can be adapted to model such workloads.

Do Edge Micro-Clusters require new management techniques? Existing research highlights the complexity and difficulty of resource management at the edge in comparison to the centralized cloud environments. This is due to heterogeneity, dynamicity and uncertainty at the network edge [8]–[10]. It is important to characterize and obtain deep insight of various edge resource management approaches.

For practical applicability, we advocate representing and studying resource management in physical, realistic edge environments, rather than using simulation-based tools. Simulations provide an abstraction of the computing environment, which may not be representative of the full scope of uncertainties [11], [12]. In this paper, we describe a testbed which represents a concrete instantiation of the HEμC concept. We provide a lightweight resource management layer which executes at the edge, implementing task allocation among the various nodes. We study several allocation strategies, from randomized allocation, through heuristic approaches, to mathematical optimization by integer programming. We show that, in a HEμC, effective task allocation is important and can cause significant improvements in overall performance. We also demonstrate that, for the scale of scheduling we perform, it is *entirely feasible* to run integer programming at the edge with minimal overhead (worst case less than 2% of the overall makespan time in our experiments).

This paper makes the following key contributions:

- We characterize and motivate the concept of heterogeneous edge micro-clusters (HEμCs).
- We describe and demonstrate typical workloads for HEμC compute platforms.
- We present an empirical study which shows that mathematical optimization is an appropriate and highly effective technique for batched task allocation in HEμCs.

The reminder of this paper is organized as follows. Section II outlines the notion of HEμCs and presents relevant use cases. Section III describes our experimental framework. Section IV presents the implemented resource allocation approaches with results analysed in Section V. The related work is reviewed in Section VI. Finally Section VII concludes the paper and considers future work.

II. Motivation

A. Heterogeneous Edge Micro-Clusters

In this section we outline the concept of a heterogeneous edge micro-cluster (HEμC). We argue this is a compelling demonstrator for a typical edge compute platform at present and for the short-term future.

A HEμC is **heterogeneous**. It will consist of multiple compute devices that have differing capabilities. This might be in terms of CPU core count and clock speed, or in terms of available RAM. Further, some devices might feature specialized accelerator nodes, such as the Movidius Neural Compute Stick. Since a HEμC is likely to be deployed for a long time, it is possible that upgraded nodes with higher specifications may be added at a later date, increasing heterogeneity in the cluster.

A HEμC is located at the network **edge**. For this reason, it is end-user facing, required to provide compute resource and serve content directly to consumers. This addresses the expectation of minimal latency. In terms of connectivity, nodes in the cluster will have a regular, non-exotic network topology, typically peers on a single LAN.

A HEμC is **micro**-scale. This has implications for its power-draw; it is likely to run off renewable energy and a battery pack. In terms of cost, the electronic equipment must be commodity, cheap components since they are so widely deployed. Further, in terms of physical footprint the HEμC should be small, perhaps occupying a road-side cabinet or embedded in the fabric of a building.

A HEμC is a compute **cluster**, consisting of a small number of nodes (perhaps up to 10) with a small number of total cores (perhaps up to 100). The multiplicity of nodes is important for (a) redundancy, given the commodity nature of the devices; and (b) parallel throughput, given the nature of the workloads that we consider in more detail below.

B. Edge Processing Workloads

In general, research literature in the field of edge computing describes high-level application use cases. Popular examples include:

1) Surveillance cameras in smart cities [13]
2) Smart home sensors [14]
3) Support for autonomous vehicles [15]
4) Voice-based smart services [7]

For our pragmatic empirical characterization approach, we prefer to identify realistic benchmark-based compute workloads that we can execute directly on HEμC nodes and report meaningful performance metrics. The DeFog benchmark suite [6] is ideally suited to our purpose. It features:

1) a representative set of pre-configured, containerized workloads with a straightforward script-based deployment model.
2) benchmarks that take common file formats (e.g. JPEGs, WAVs) as input.
3) realistic small-scale tasks that could be aggregated to produce large-scale use cases such as the edge applications outlined above.

There is a key difference in how we use the DeFog benchmarks. The original developers compare execution of each individual task across three deployment platforms: cloud, edge or fog. We propose to focus on edge deployment and use the benchmarks to study batched execution of multiple concurrent tasks on multiple nodes in a heterogeneous edge environment.

Others have identified this trend of heterogeneity at the edge [10]. We therefore aim to tackle the lack of a real testbed for HEμC and characterise its deployment features and utility by modeling and experimenting with a real configured testbed instead of using simulation.

III. Experimental Infrastructure

A. Philosophy

The majority of studies in edge resource management rely on simulation-based tools such as CloudSim [16] and iFogSim [17]. We prefer an empirico-realist approach to evaluation, which requires the use of physical testbed hardware, representative benchmark applications software, and wall-clock timing metrics for performance reporting.

While we acknowledge the scale of our work is necessarily limited by these realism constraints, we feel our findings will be more tangible and translatable to near-term, pragmatic edge deployment scenarios.

B. System Testbed

Our testbed is a minimal instantiation of a HEμC, featuring heterogeneous resources that are capable of processing typical edge computation and storing relevant data.

As this framework is a distributed system, we anticipate that there is an intelligent resource management component that maintains a profile of each node (CPU load, memory, network

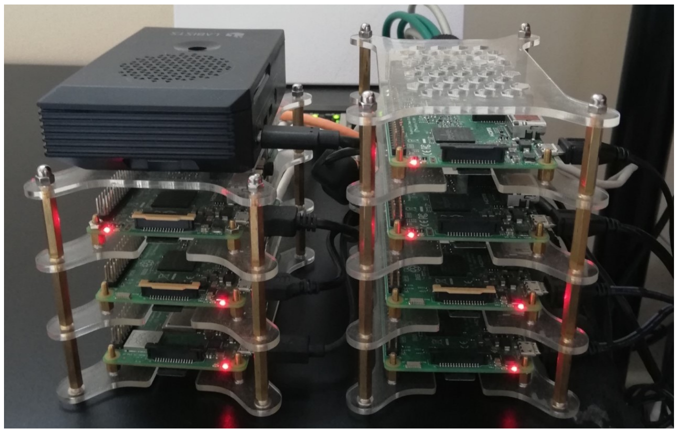
Fig. 1. Prototype heterogeneous edge micro-cluster with Raspberry Pi nodes

TABLE I
RASPBERRY PI DEVICES SPECIFICATIONS: OUR 4-NODE CLUSTER COMPRISES ONE OF EACH MODEL, OUR 8-NODE CLUSTER COMPRISES THREE 2BS, THREE 3BS AND ONE OF EACH OTHER MODEL

model	max GHz	core count	RAM GB
2B	0.9	4	1
3B	1.2	4	1
3B+	1.4	4	1
4B	1.5	4	4

bandwidth, etc.), and creates appropriate allocation plans to meet QoS requirements in each application scenario.

Our edge cluster comprises eight single-board computers (SBC) that represent heterogeneous edge nodes, with 32 available compute cores. SBCs are drawn from different generations of the Raspberry Pi device. Our edge testbeds are limited to Raspberry Pi devices; however, in the principle, we could include comparable SBCs such as Odroid or BeagleBone. Table I presents the specific Raspberry Pi devices used in this paper.

While the devices share the same Arm architecture, they are heterogeneous in that they have different chipsets with differing cache sizes and clock speeds. The nodes are connected in a flat topology with a gigabit ethernet switch. Each node is running the latest instance of the Raspbian Linux OS. Figure 1 shows the testbed setup.

This HEμC is characteristically reconfigurable since it is:

1) expandable, to accommodate a variety of devices.
2) portable, to move the testbed to another location.
3) distributable, to split the testbed across several locations.

C. Benchmarks

Blackburn et al. [18] emphasize the importance of using benchmarks that are realistic, representative and reflective of real-world scenarios. This is to enable researchers to reach valid conclusions. Therefore, based on existing work [6], we adapt the DeFog benchmark suite to generate HEμC workload scenarios. As mentioned above, DeFog originally compared the performance of cloud and edge platforms by executing independent application runs. We have extended the benchmark harness to run concurrent workloads across multiple edge nodes. Three DeFog application scenarios were re-targeted in this paper:

1) Object detection (YOLO): This application deploys deep learning to provide a real-time object classification for image processing. The application receives image files in .jpg format, runs a pre-trained neural network model to provide an estimation of the objects inside the image, and then returns the result files as image files with overlaid object classifications and corresponding confidence level.
2) Speech-to-text conversion (PocketSphinx): This application converts audio files to text files. The application receives audio files in .wav format and uses a pre-trained model to generate .txt files that contain the recognized text.
3) Text-audio synchronization (Aeneas): This forced alignment application works to automatically generate a synchronization map between text fragments and audio clips. This application takes paired audio files and text files as input (.wav and.txt) and generates textual output files with embedded timing metadata, e.g. for closed captioning.

These particular tasks are likely to become increasingly popular for end-users given the growing need for environment awareness and digital accessibility.

Since the original DeFog framework only supplied limited input data for each benchmark, we have added supplementary input data sets. This allows us to observe batch mode execution of multiple concurrent instances of an application with different inputs. Effectively, we are modeling task-level parallelism for these applications across a cluster of heterogeneous nodes.

IV. RESOURCE ALLOCATION APPROACHES

As resources in edge micro-clusters are limited, they need to be utilized effectively and efficiently. Such edge clusters should be configured to be 'self-managed', since human intervention is rarely available. Workloads are expected to be heterogeneous and highly dynamic. Thus, edge clusters should be able to self-adapt to deal with workload variations.

Our lightweight HEμC management monitors resources at each node in the cluster. This simply polls the Linux OS in each node for appropriate metric values like `/proc/loadavg`. In principle, this monitoring could run on a democratically elected head node of the cluster. For our profiling experiments, the monitoring was done on a Macbook connected to the HEμC LAN. This section discusses the resource allocation strategies we considered in our study.

We want to place the compute tasks on the most eligible node according to some criteria, by generating a deployment plan based on the current condition of the cluster. The main objective is to minimize the total makespan of the tasks, by efficiently using the resources of the HEμC. We implemented and evaluated four resource allocation approaches, as outlined below.

A. *Cluster Election-based*

The cluster-election method periodically monitors nodes' CPU load averages and nominates a potential node to perform any incoming task. We only considered nodes' CPU load average as a node nomination criteria. This method could be further enhanced by integrating other criteria, such as the nodes' power level or nodes connections. Algorithm-1 presents pseudocode of cluster-election resource allocation algorithm.

B. *Best-Node Selection*

In this resource allocation approach, the incoming tasks are always allocated to the node that has the best static resource specification, which is Raspberry Pi-4 in our micro-cluster.

C. *Randomized allocation*

The incoming tasks are placed randomly on an arbitrary node without any resource or capacity consideration. All nodes are equally likely to be targeted.

D. *Mixed-Integer Programming (MIP) allocation*

The above two greedy heuristic methods show that when nodes become overloaded, processing powers decrease and tasks' execution times increase. In this method, we consider balancing the workloads between the cluster's nodes, taking into account both nodes' processing capacities and the workloads' requirements. The problem is formulated as a combinatorial assignment problem and solved using mixed-integer programming. MIP allocates tasks based on a pre-estimated cost matrix that estimates the amount of time each task might need on each node and considers nodes' processing capacities, how many tasks that each node can process simultaneously without consuming its resources. The resource manager in this approach forwards the incoming tasks based on the mapping plan produced by the MIP allocation solver. Section IV-D1 presents the problem formulation and defines related constraints. We implemented this approach using OR-Tools provided by Google [19].

Algorithm 1: Greedy Cluster Election

```
Input: $T$ (set of offloaded tasks);
$N$ (set of nodes in edge cluster)
for task $t$ in $T$ do
    for node $n$ in $N$ do
        measure current CPU load of $n$;
        let $n'$ be the node in $N$ with lowest CPU load;
    end
    Assign $t$ to $n'$;
    Commence immediate execution of $t$;
end
```

1) Problem Formulation: Based on the introduced specifications of our HEμC in section III-B, and the specifications of the workloads defined in section III-C, we formulate the problem as an assignment problem of a set of independent tasks in a heterogeneous edge micro-cluster. The aim is to make a decision plan that maps each task t on a suitable processing node n in our HEμC.

TABLE II
PROBLEM NOTATIONS

Notation	Meaning
T	set of tasks to be allocated
N	set of cluster nodes
t	individual task
n	individual node
x_{ij}	decision variable (*indicating whether node i executes task j*)
cap_n	processing capacity of node n
$cost[i][j]$	cost matrix (*indicating cost of node i executing task j*)

The overall objective is to find an efficient assignment plan that efficiently utilizes the cluster's nodes such that all tasks are assigned to at least one cluster's node (*constraint-1*) and to maintain the upper bound capacity of each cluster's node (*constraint-2*). Meanwhile, the total makespan time to complete all tasks is minimized. Table II outlines the notations used in the problem.

Given: a set of independent tasks T in which each task has an estimated cost that represents how much time a task t requires on each processing node n; and a set of cluster's processing nodes N with varying size capacities and each node has an upper bound. The objective is to produce an efficient mapping plan of tasks on cluster nodes in HEμC such that the total makespan time is minimized.

Decision variables: we consider a decision variable x_{nt} as a binary integer variable 0 or 1, which denotes whether a cluster node n is assigned to a task t or not.

$$x_{nt} = \begin{cases} 1, \text{if node } n \text{ allocated to task } t \\ 0, \text{otherwise} \end{cases} \tag{1}$$

Constraints: we consider two main constraints respecting our edge micro-cluster's heterogeneity and resource constraints:

1) *constraint-1:* we introduced a node capacity constraint cap_n, in which each cluster node has a maximum capacity. The total number of tasks allocated to each node should not exceed the upper bound capacity of the node. Otherwise the node's compute resources will become saturated, and the task execution time will be prolonged.

$$\left(\sum_{t \in T} x_{nt}\right) <= cap_n \quad , \quad \forall \;\; n \in N \tag{2}$$

2) *constraint-2:* each task t must be allocated to at least one node. This is to make sure that all tasks are assigned.

$$\sum_{n \in N} x_{nt} >= 1 \quad , \quad \forall \;\; t \in T \tag{3}$$

Objective: the objective is to minimize the overall makespan time of a set of tasks while considering resource constraints of our HEμC. Thus, the decision is where to place a task t to a node n such that we minimize the overall makespan time of a set of tasks and respect upper limit of the capacity nodes.

$$\text{Minimize} \sum_{n \in N} \sum_{t \in T} cost[n][t] * x_{nt} \quad (4)$$

V. Performance Evaluation

A. Minimizing Makespan

The empirical evaluation considers the makespan time to complete a batch of benchmark invocations based on the different allocation approaches.

The *makespan* in our study is defined to be the wallclock time for the edge micro-cluster to complete processing a set of tasks, defined from the time when the first request is generated to the time that all cluster's nodes complete processing their assigned tasks. In particular, the makespan time includes: (1) *the communication time* $\boldsymbol{C}$, which is the time to offload assets to the cluster's nodes and to receive final results back, (2) *the decision time* $\boldsymbol{D}$, which is the time to compute the assignment decision in the MIP-allocation approach or the time to decide a node in the greedy approaches, the cluster election and best node selection, and (3) *the execution time* $\boldsymbol{E}$, which is the actual time each node takes to process the assigned tasks.

We run the experiments on two edge micro-cluster configurations. Each cluster comprises of a set of heterogeneous nodes. The first cluster contains four nodes, and we expand the nodes to eight in the second cluster. Table I shows the specifications and the quantity of the devices deployed in each cluster.

As mentioned above, the overall objective is to minimize the batch execution makespan time, measured as wallclock-time, while efficiently utilizing the resources in our HEμCs. We examined the performance of the resource allocation methods, explained in section IV, on three DeFog benchmarks, the image-detection, audio-to-text converting, and audio-text synchronisation, respectively. We measure the makespan time to process 32 concurrent tasks for each benchmarks.

Figures 2 and 3 show the makespan time for the 4-nodes cluster and 8-nodes cluster, respectively. The graphs show results based on the mean of 15 runs. It is clear that the MIP-based approach efficiently utilizes clusters' resources and optimises the benchmark batch execution for computation-intensive applications, while it achieves a comparable performance to other greedy approaches for the audio-text synchronisation benchmark. The two greedy heuristics methods, cluster-election and the best-node selection, achieve comparable performance in all benchmarks. The cluster-election method successfully reflects the nodes CPU load averages and efficiently elect nodes. The random-based allocation clearly incurs high execution times for all benchmarks.

Overall, the MIP approach effectively represents the tasks' computation times and the nodes' capacities. It successfully optimises the makespan processing time and outperforms other approaches for computation-intensive applications, i.e., image-detection and audio-to-text converting. In addition, the MIP approach shows a stable performance when cluster nodes increase. The cluster-election approach demonstrates a good performance to reflect the clusters' conditions and nominates potentially capable nodes. The cluster-election achieves a better performance than the best-node selection for computation-intensive benchmarks. However, this approach could be further enhanced by integrating other nomination criteria (e.g., nodes connections, power level for clusters that run on renewable energy, etc). The selection of the best node is not always an intelligent choice for the edge-micro clusters, as nodes processing time increase when overloaded. Finally, for light-computation applications (i.e., audio-text synchronisation), the MIP-allocation approach performs as well as the best node selection.

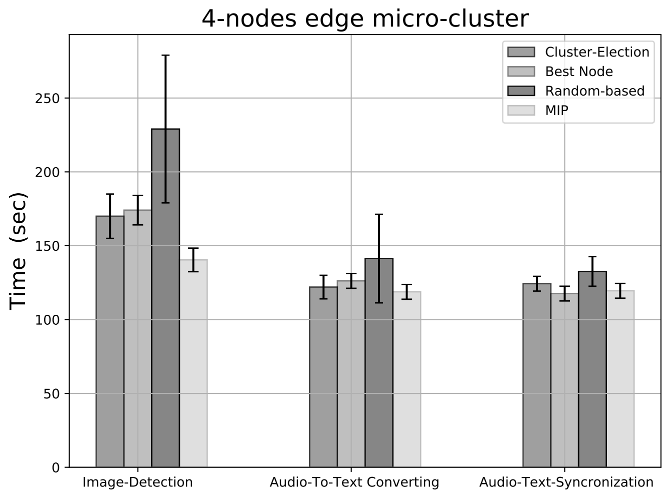

Fig. 2. Makespan of 32 batched concurrent tasks based on different allocation approaches running on 4-nodes edge micro-cluster (confidence intervals indicate one standard deviation)

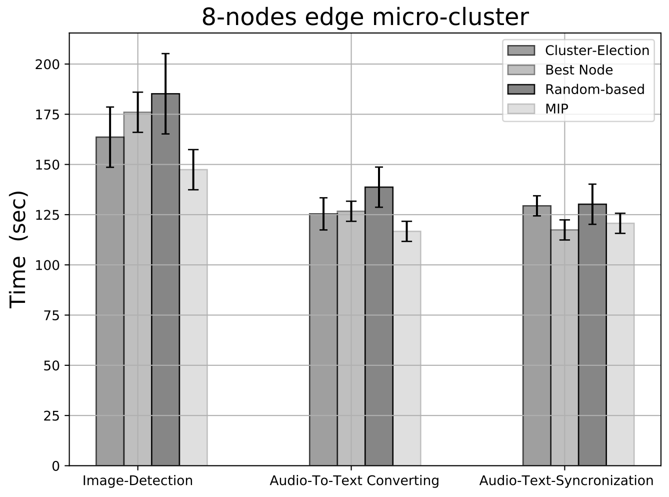

Fig. 3. Makespan of 32 batched concurrent tasks based on different allocation approaches running on 8-nodes edge micro-cluster (confidence intervals indicate one standard deviation)

B. Allocation Overhead

Allocation overhead is a critical metric in edge computing. Allocation techniques should be light, not adding excessive overhead as most IoT-based applications and management techniques are being deployed in resource-constrained devices and require lightweight management techniques. If the allocation techniques require expensive resources in terms of CPU, memory or energy, it would not be applicable in the context of edge computing [11]. Therefore, it is imperative to consider the *lightness* and the *overhead* of the technique when designing resource allocation for edge context scenarios.

All makespan times reported above *include* the time taken for task allocation calculations, i.e. solution time for the MIP approach, live metric query time for the greedy heuristics, pseudo-random function evaluation time for the random approach. We deliberately added 3 sec networking delay between sending tasks to the allocated nodes. There were two main reasons to add this delay: first, to avoid any networking bottleneck or task failures that might occur when sending several concurrent tasks to the same node, and secondly, to allow the cluster-election to update the elected node. This 3 sec delay was added to all resource allocation approaches to ensure a fair comparison.

For the MIP approach, the allocation overhead never exceeds 1 second, which in the worst case is still below 2% of overall makespan time. We observe that for our HEμC setup with these workloads, there is limited complexity for the MIP constraints meaning solution times are very short. In a limits study experiment, we discovered that MIP allocation for typical edge workloads and clusters remains below 1 second, for up to 1000 tasks (see Figure 4).

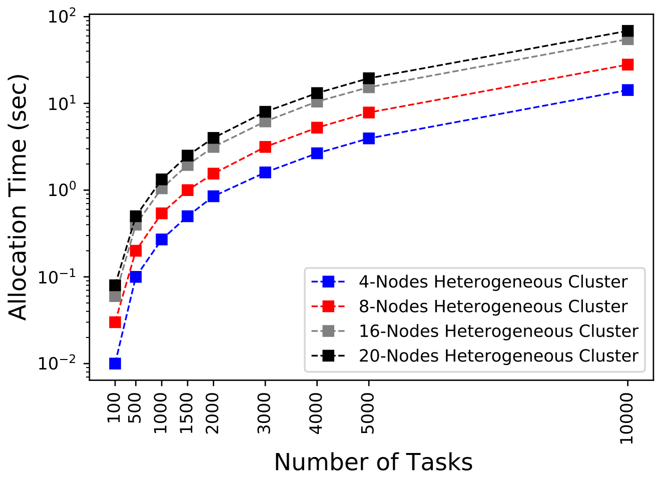

Fig. 4. Allocation Overhead for MIP Approach

C. Reflection

Overall, the experiments demonstrate that HEμCs are able to handle heterogeneous workloads in real-world edge application scenarios. This requires light and efficient resource management to orchestrate the edge nodes. The deployment of lightweight but effective resource management approaches for batched task execution is straightforward, and can yield mathematically optimal solutions. The work presents a basic comparison between a set of lightweight resource allocation methods that are capable of improving resource allocations in micro-clusters. The evaluations reveal that the mixed-integer programming resource allocation with the respect of the nodes capacities outperforms other greedy-heuristic resource allocation and efficiently utilise the cluster resources. The experimental infrastructure (the benchmarks and the cluster testbed) could be used to extend this work and to implement further research. The need for investigating other complex resource allocation approaches for edge computing will be considered in future work.

D. Limitations

Our evaluation has been conducted on two small-scale HEμC configurations and limited to the Raspberry Pi commodity, with a small set of benchmarks. The issue of generality has not been addressed fully and will be the subject of future work. The study was limited to optimise the makespan metric while other performance metrics, e.g, operational cost and power consumption, were reserved for future work. Finally, the correlation between makespan time and the benefit of expanding cluster's nodes (scalability and elasticity) is an area for future investigation.

VI. Related Work

Resource management is a critical concern in edge computing [9], [10], [20]. The application placement concept covers approaches and methodologies to find feasible solutions for mapping between IoT-based application workloads and the computational resources [9], [10], [21]–[25].

However, the majority of studies on edge resource management are implemented and evaluated using simulators, such as CloudSim [16], iFogSim [17], or Matlab [9]. We argue that it is preferable to investigate resource management in real testbed environments.

Recent studies demonstrate the feasibility and suitability of using single-board computers (SBCs) to build edge clusters to perform intensive computational tasks instead of cloud data centers. Johnston et al. [26] consider SBC clusters to be "the game-changer" in pushing computation from centralized cloud data centers to decentralized edge data centers.

Integer programming seems to be an inappropriate technique for resource management in cloud, since there is high complexity when the problem size increases [27]–[30]. Therefore, most studies implement heuristics-based or metaheuristics-based solutions for Cloud Computing scenarios [12]. In this paper, we argue that integer programming is an efficient and effective method to find the optimal task allocation for HEμCs where the problem size is tractable.

VII. Conclusions and Future Work

The HEμC design concept has the potential to make edge computing more affordable and more accessible to everyday end-users and smart citizens [31]. It is likely that such HEμCs will comprise commodity compute devices similar to Raspberry Pi nodes.

However, resource management is critical for effective edge computing. In this study, we have presented and characterized an approach to resource management for HEμCs based on batch arrival of containerized edge workloads. We have demonstrated empirically that such an execution paradigm is highly amenable to mathematically optimized resource management using integer programming, with this approach giving optimal or near-optimal makespan performance even when the overhead of integer programming is included.

Our study was limited to task allocation in a small-scale cluster. However, this work can be extended in future by:

1) expanding the testbed to include more nodes.
2) exploring other resource management aspects such as load balancing and task migration.
3) evaluating more complex techniques such as multi-heuristic-based or meta-heuristic optimization approaches.
4) considering other relative metrics such as power consumption and operational cost.

Acknowledgments

This work is partly supported and funded by King Khalid University, Saudi Arabia and the Saudi Arabian Cultural Bureau in the UK.

References

[1] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[2] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[3] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.

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

[5] Justyna Winkowska, Danuta Szpilko, and Sonja Pejić. Smart city concept in the light of the literature review. *Engineering Management in Production and Services*, 11(2):70–86, 2019.

[6] Jonathan McChesney, Nan Wang, Ashish Tanwer, Eyal de Lara, and Blesson Varghese. Defog: fog computing benchmarks. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 47–58, 2019.

[7] Anirban Das, Stacy Patterson, and Mike Wittie. Edgebench: Benchmarking edge computing platforms. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 175–180. IEEE, 2018.

[8] Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 20–26. IEEE, 2016.

[9] Mostafa Ghobaei-Arani, Alireza Souri, and Ali A Rahmanian. Resource management approaches in fog computing: a comprehensive review. *Journal of Grid Computing*, pages 1–42, 2019.

[10] Cheol-Ho Hong and Blesson Varghese. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys (CSUR)*, 52(5):1–37, 2019.

[11] Mohammad S Aslanpour, Sukhpal Singh Gill, and Adel N Toosi. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, page 100273, 2020.

[12] Yaser Mansouri and M Ali Babar. A review of edge computing: Features and resource virtualization. *Journal of Parallel and Distributed Computing*, 2021.

[13] N. Chen, Y. Chen, E. Blasch, H. Ling, Y. You, and X. Ye. Enabling smart urban surveillance at the edge. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 109–119, 2017.

[14] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[15] Thomas Rausch, Clemens Lachner, Pantelis A Frangoudis, Philipp Raith, and Schahram Dustdar. Synthesizing plausible infrastructure configurations for evaluating edge computing systems. In *3rd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 20)*, 2020.

[16] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[17] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[18] Stephen M Blackburn, Kathryn S McKinley, Robin Garner, Chris Hoffmann, Asjad M Khan, Rotem Bentzur, Amer Diwan, Daniel Feinberg, Daniel Frampton, Samuel Z Guyer, et al. Wake up and smell the coffee: evaluation methodology for the 21st century. *Communications of the ACM*, 51(8):83–89, 2008.

[19] Laurent Perron and Vincent Furnon. OR-Tools.

[20] Sukhpal Singh and Inderveer Chana. A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of grid computing*, 14(2):217–264, 2016.

[21] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Optimized iot service placement in the fog. *Service Oriented Computing and Applications*, 11(4):427–443, 2017.

[22] Mohit Taneja and Alan Davy. Resource aware placement of iot application modules in fog-cloud computing paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1222–1228. IEEE, 2017.

[23] Salvatore Venticinque and Alba Amato. A methodology for deployment of iot application in fog. *Journal of Ambient Intelligence and Humanized Computing*, 10(5):1955–1976, 2019.

[24] Nan Wang and Blesson Varghese. Context-aware distribution of fog applications using deep reinforcement learning. *arXiv preprint arXiv:2001.09228*, 2020.

[25] Shiqiang Wang, Rahul Urgaonkar, Ting He, Kevin Chan, Murtaza Zafer, and Kin K Leung. Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1002–1016, 2016.

[26] Steven J Johnston, Philip J Basford, Colin S Perkins, Herry Herry, Fung Po Tso, Dimitrios Pezaros, Robert D Mullins, Eiko Yoneki, Simon J Cox, and Jeremy Singer. Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, 89:201–212, 2018.

[27] Zhen Ye, Xiaofang Zhou, and Athman Bouguettaya. Genetic algorithm based qos-aware service compositions in cloud computing. In *International Conference on Database Systems for Advanced Applications*, pages 321–334. Springer, 2011.

[28] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Computing Surveys (CSUR)*, 47(4):1–33, 2015.

[29] Qiang Li and Yike Guo. Optimization of resource scheduling in cloud computing. In *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 315–320. IEEE, 2010.

[30] Yi Wang, Ye Xia, and Shigang Chen. Using integer programming for workflow scheduling in the cloud. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 138–146. IEEE, 2017.

[31] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.