Fog Resource Selection Using Historical Executions

Nour Mostafa, Ismaeel Al Ridhawi, Moayad Aloqaily
College of Engineering and Technology
American University of the Middle East (AUM), Egaila, Kuwait
{Nour.Moustafa, Ismaeel.Al-Ridhawi, Moayad.Aloqaily}@aum.edu.kw

Abstract— **As an emergent technology, IoT promises to harness the computational and storage resources distributed across different remote clouds. Fog computing extends cloud computing by bringing the network and cloud resources closer to the network edge. Those resources are typically heterogeneous in nature requiring careful management. As the number of resources contributing to a cloud grows, so the problems associated with efficient and effective resource selection and allocation increase. In this paper, we introduce a fog resource selection algorithm (FResS) that enables automated fog selection and allocation for IoT systems. The proposed model collects and maintains a repository of fog performance data in the form of execution logs stored in standard formats. When a new task needs to be executed, prediction for its run-time is made by using these logs, which results in a realistic run-time estimate as well as best fog selection for task placement. Simulation results show a decrease in the overall end-to-end (e2e) latency of the system.**

Keywords—cloud, fog, fog-to-cloud, fog-to-fog, e2e delay.

I. Introduction

Cloud computing has provided a centralized solution to resource-constraint end-user devices. Computing, storage, and communication resources available at the cloud and advances in telecommunication networks has made it possible for most end-user devices to have increased mobility functionalities. This is achieved by limiting the tasks for end-user devices to sensing and limited processing. For complex and data-constraint task, mobile devices simply forward the acquired data to the cloud for further processing. Fog computing came to mitigate the shortcomings of the cloud computing scheme within the IoT environment by bringing network, processing, and storage resources closer to end-user devices, hence helping them meet their hard-constraint requirements and offloading much of their data to the fog. Additionally, communication between the fog and cloud (F2C) was made possible allowing jobs to be executed either at the fog or cloud layer depending on the requirements of the job request [1].

Fog systems are expected to scale up, where in such situations it is easy to see that the overhead incurred in executing tasks are also increasing and will continue to grow. Consequently, the increased complexity of fog systems in terms of number of users, resources, and number of tasks, results in increased time for task executions. To solve this issue, a new solution was proposed in [2] to allow fogs to communicate with each other to process task requests. The solution was referred to as Fog-to-Fog (F2F) communication. In such systems, fogs are not limited to either execute a task or forward it to the cloud, but also to communicate with other fogs to process the job request, hence minimizing the overall end-to-end latency.

One part of the F2F communication scheme that was neglected is how fog resources are selected. When a job request requires certain fog or cloud resources for it to be executed, it is difficult to determine the number of resources required as well as the best fog to process the request. In this paper, we provide an extension to the work introduced in [2], where a fog resource selection algorithm (FResS) is integrated within the IoT framework. The selection and location of the best fog to execute the task requires some form of assistance to enable fog selection to be determined in advance to avoid incurring overhead when executing the task on resource-limited fogs. The FResS technique stores historical data related to users and devices creating execution logs to be used in predicting the time and number of resources needed to complete a new task. The introduced solution provides fast and accurate execution time predictions, resulting in a decrease of the overall e2e latency of the system.

The rest of the paper is organized as follows: Section II provides a summary of the related work in the literature. Section III introduces the FResS module. Section IV provides some of the data gathered from simulation results. Finally, Section V concludes the paper and provides further research directions for future work.

II. Related Work

Studies on fog computing are still ongoing. In [3] the Fog to Cloud (F2C) architecture was first introduced that split the cloud and fog architecture into layers of cloud and fog. The authors proposed a management system responsible for discovering a set of available fogs and then choosing the best fog that meets the task requirements. The solution assumes that the requested task can be divided into individual functions which in turn are split into different fog layers. The presented strategy did not show or explain how services are divided into individual functions, and the criteria considered for selecting the most optimal fog for the task.

The authors in [1] proposed a cloud solution that aims at achieving low delay on service allocation by using service atomization which is responsible for splitting the service into distinct slices, tailored to enable parallel execution. The atomic services will be distributed among the available F2C resources by matching atomic service requirements and F2C resource offerings. The authors did not take into consideration resource capabilities such as processing power and speed.

In [4] the authors introduced a prediction method for cloud computing resource consumption. The solution classifies cloud services to guarantee near accurate predictions. Additionally, machine learning algorithms are adopted within their solution to predict future number of requests. When predicting the resource amount consumed to complete the task for an incoming request, a placement algorithm is used to allocate it to the best processing unit. The prediction method further allows sensing the resource usage in cases when new hardware is introduced to the system. An iterative correction process is used to compare the measured and predicted total resource consumption to increase prediction precision for future tasks.

The authors in [5] proposed a solution that uses graph partitioning theory to achieve load balancing among fogs. A system model of fog computing is constructed by combining the graph theory and characteristics of fog computing. Graph repartitioning is considered to achieve a dynamic load balancing mechanism. Another solution presented in [6] considered optimal workload allocation in Fog-Cloud computing systems. A systematic framework is developed to investigate the power consumption-delay tradeoff issue in F2C scenarios. The problem is divided into three approximate sub-problems, namely, 1) power consumption-delay tradeoff for fog solved using convex optimization techniques, 2) power consumption-delay tradeoff for cloud solved using a mixed integer non-linear programming problem, and 3) minimizing communication delay using the Hungarian method in polynomial time [7].

Although work is progressing in solving issues related to fog computing. The current related literature lacks in solving the problem of resource selection for fog computing. As mentioned earlier, this work considers a solution to select fog resources based on the required task execution time and previous results of successful resource selection runs.

III. Fog-Prediction and Selection Scheme

The overall purpose of this paper is to present introduce run-time predictions within fog environments for resource selection, leading to the development of a run-time prediction model. The work presented in this paper is divided into two parts: i) the development of a fog execution time prediction model, and ii) the development of a fog resource selection model. We introduce a new module called Fog Resource Selection (FResS) in the fog layer to manage request from IoT devices towards fog entities. It can be seen from Figure 1 that the FResS module incorporates a Task Scheduler, Resource Selector, and History Analyzer. These modules perform different tasks related to resource selection, predictions and history management.

A. Task Scheduler

The Task Scheduler sits between IoT devices and the Fog layer. It provides an interface to the IoT devices, Fog, and Cloud layers. When a new task arrives, it is forwarded to the Task Scheduler Module. The Task Scheduler module sends the task description to the History Analyzer to find similar tasks to predict the execution time and the required resources for

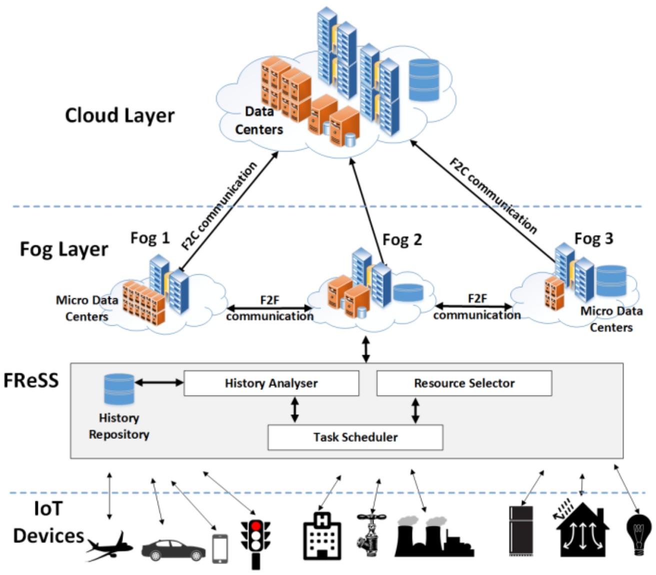

Fig.1. Incorporating the Fog Resource Selection (FResS) module within the F2C architecture.

executing the task. Once related tasks are found, the next step is to make a run-time prediction of the number of resources required and the time needed to complete the task. The Resource Selector identifies the fogs that are able to complete the task and the fog resources to be used. After the resource selection is complete, the Resource Selector returns a list of the selected resources depending on the configuration, along with the execution time for each task to the Task Scheduler Module. The Task Scheduler module in essence forwards the execution log to the History Analyzer to be used for future task requests. Figure 2 outlines the described process through a sequence diagram, showcasing the sequence of interaction considered between different FResS components.

The Resource Selector component of the proposed solution follows a multi-step process. In the first step, fog resources are discovered and tested against the essential minimum criteria,

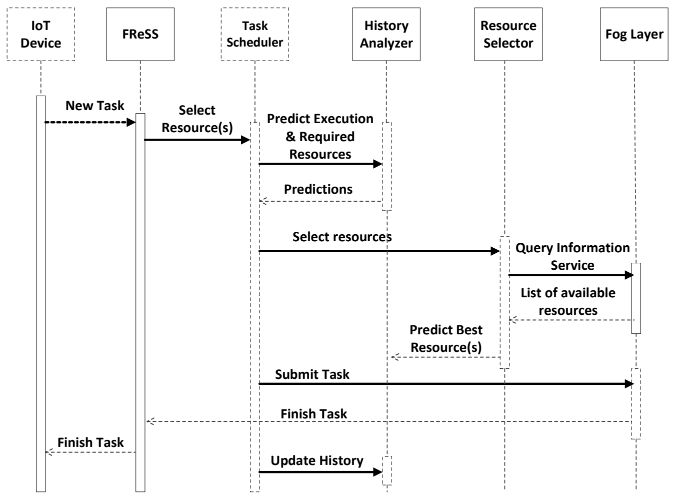

Fig.2. Sequence diagram showing the interaction between different components of the FResS module.

e.g. availability, hardware architecture, operating system, software libraries, memory, and cost. All of the resources selected after step one are eligible to execute the current task if there are no deadline constraints. Step two evaluates resources in terms of their ability to meet deadlines by evaluating their predicted execution time. These predictions are then utilized for optimum task placement by the Task Scheduler. In our future work, we will be considering additional constraints such as cost, data locality, workflow constraints, co-allocation, advance reservation, and user preferences to assure optimum resource selections are achieved. These additional constraints should be evaluated by the Task Scheduler after receiving a ranked list of resources along with predictions from the Resource Selector. Step two is facilitated by the availability of predictions. Predictions play a key role by providing the possible outcomes of a certain resource selection. Predictions helps the Task Scheduler to decide about the required duration of advanced resource reservations. The co-allocation decision is facilitated with this predicted knowledge of expected execution times.

The proposed Task Scheduler module is very simple in the sense that all the external constraints of cost, and data locality were ignored. It provided a more focused approach to the evaluation of the performance of resource selection, using predictions. In a real fog scenario, all these constraints must be considered. Additional information about the expected execution times will make it easier to deal with these additional constraints.

B. History Analyzer Module

The History Analyzer module is responsible for examining through the history data to generate run-time predictions. The Task Scheduler module forwards the input task description to this module, then a search through the historical execution logs is made to find similar tasks. Such a learning-based method is conducted using Artificial Neural Networks (ANNs) [8].

ANNs can work with hundreds or even thousands of passes of data sets [9][10]. Fog/cloud computing deals with such very large number of resources. ANNs generally consist of three layers, input, middle (or hidden), and output [9]. Data enters the system at the input layer, referred to as the starting point. The hidden layer is the intermediate processing unit at which the input data is passed to for processing. Finally, the processed data is then passed onto the output layer. The key element of processing information in neural networks is its interconnecting weights. Weights express the relative strength of the input data or the various connections that transfer data from layer to layer. The network learns through repeated adjustment of weights until actual and desired output is sufficiently accurate for training to stop. The process of adjusting weights during the learning process is referred to as the learning algorithm.

Once the Task Scheduler finds similar task using the historical execution logs, execution time predictions are generated for all resources in the list. The complete list is then returned to the Resource Selection module. Algorithm 1 determines the required resources needed for a task to execute. The resource identification process is dependent on five

Algorithm 1: Task Execution Time Predition

```
Data: Task log history
Input: New task submission description file
Result: Execution time predictions for the chosen resource
start
  while taskset <> empty
  do
    Process request for task$i$;
    while for all tasks logs
    do
      NN predict task execution time and required resources;
      add resource to the list;
    end
    select next task;
  end
end
return finished required resource list;
```

parameters: delay, price, previous execution time, memory requested, and submission time. It has been noted that many users of fog networks have a tendency to be repeatedly doing the same tasks and using the same data [9][11]. This creates an opportunity to develop the proposed model to use these parameters which are actually used to predict the execution time and required resources of the new task from execution logs. These execution logs can then be used to generate a prediction for the new tasks.

After every execution, results are stored as an execution history log file and subsequently used as training vectors for the ANN to make predictions for future tasks. As the number of historical executions increases it is expected that accuracy of predictions will also increase [12][13]. Using the proposed prediction model, the History Analyzer is able to determine the required resources in one step and the result is immediately returned to Task Scheduler.

The History Analyzer estimates the execution time and required resources of the task with sufficient accuracy. Based on these predicted results, the Resources Selector uses this information to determine the best resources to execute the task. The proposed modules deal with improving the performance issue of IoT devices that communicate directly (or randomly) with the remote fog/cloud for tasks submission which incurs high delays and a network overload.

The prediction results generated are stored in a separate database called the Prediction Model DataBases (PMDB). For a new task, the PMDB is searched, if a similar task is in the database, the result will be sent back. After a task is completed, the parameters are stored in the PMDB for future executions. If not, F2F communication will take place to execute the task.

The current fog computing solution treats each task as a new task without taking into consideration any feedback from past history executions. The proposed prediction model on the other hand uses history to inform the system about incoming queries, and the database is updated constantly. Therefore, the prediction model database does not have static parameters, it changes based on the new task parameters which help provide more accurate and efficient predictions.

C. *Resource Selector Module*

The Resource Selector Module is responsible for selecting resources that are best suitable for the incoming task. It receives input from the Task Scheduler module which is generated by the History Analyzer module. Depending on the implementation of the Task Scheduler it can provide a single resource or a list of resources. The proposed module will provide the best suitable resource(s). If a list of resources along with predictions are returned, then the final resource selection is carried out by the Task Scheduler by considering other constraints such as cost and deadlines.

IV. SIMULATION RESULTS

To show the advantages of using the fog resource selection method, we simulated the fog and cloud environment using GridSim [14]. A comparison of the proposed F2F solution against a traditional cloud solution is performed. We assume that IoT devices reside on geographically distributed sites and model the network as a graph of nodes that represent storage sites in the fogs. Nodes are assumed to have a uniform bandwidth, computing power, memory and storage capacity. We simulate different scenarios by varying the number of files, size of files, number of job requests and capacity of storage nodes. The simulated scenarios assumed the presence of one cloud storage site, three fog storage sites, and 150 IoT devices. Files sizes range between 100 GB and 500 TB. Bandwidth connectivity is up to 2000 MB/Sec. The total number of job requests is 1500.

Four different file access requests are used to compare the two techniques: small, medium, large and very large file size requests. Simulation results show a decrease in file access delay. Figure 3 shows that the average response time required to gain access to a single stored file using the FResS solution is reduced by up to 46% for large file sizes when compared to the traditional cloud files access technique. Cloud files are replicated to multiple fogs rather than a single cloud storage.

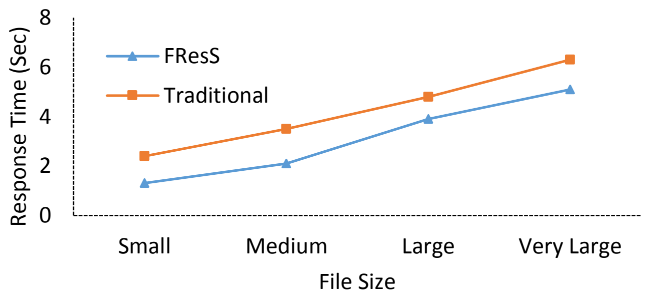

Fig.3. Average response time for a single file access with varying sizes.

The same experiment is repeated while varying the number of requested small size files. Results depicted in Figure 4 show that access time when using the FResS technique outperforms the traditional technique in all file request cases with up to 42% reduction in average response time.

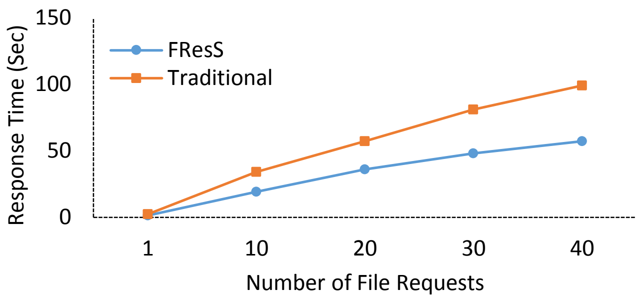

Fig.4. Average response time for multiple small size files access requests.

Results depicted in Figure 5 show that access time for multiple large files when using the FResS technique outperforms the traditional technique in all file request cases with up to 45% reduction in average response time.

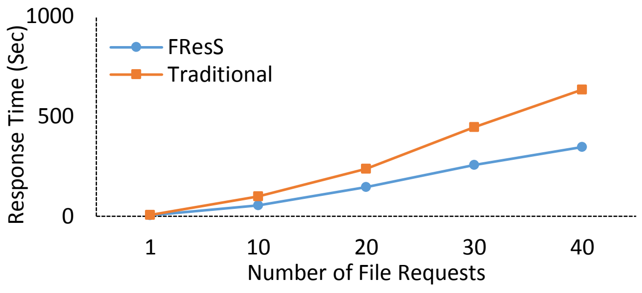

Fig.5. Average response time for multiple large size files access requests.

V. CONCLUSION AND FUTURE WORK

The recent advances in IoT technologies have led to the introduction of Fog Computing. Fog Computing is surely an interesting solution for many enterprises and businesses which include computation, storage, and data exchange capabilities of edge devices. However, it became clear that the present fog architecture will not be able to handle the enormous amounts of tasks submitted by thousands of nodes across the network when seen in the IoT context. In this paper, we introduced a novel approach in fog computing, namely, a F2F resource selection algorithm called FResS, that allows for automated fog selection and allocation to IoT device job requests. The selection and allocation of the best fog to execute the task allows for a decrease in the overhead incurred when executing a task on resource-limited fogs. The proposed FResS module consists of three components: Task Scheduler, Resource Selector, and History Analyzer. These components perform different tasks related to resource selection, execution time predictions and history management. The work presented here could still be improved by examining more QoS parameters that could affect the fog selection process. Our future work will target validating our model through implementation.

REFERENCES

[1] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez and S. Sanchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios," in Proc. 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, 2016, pp. 1-6.

[2] W. Masri, I. Al Ridhawi, N. Moustafa, P. Pourghomi, "Minimizing Delay in IoT Systems Through Collaborative Fog-to-Fog (F2F)

Communication," the 9th International Conference on Ubiquitous and Future Networks (ICUFN 2017).

[3] X. Masip-Bruin, E. Marín-Tordera, G.Tashakor, A. Jukan, G.Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," the IEEE Wireless Commun. 23(5): 120-128 (2016).

[4] F. Derakhshan, H. Roessler, P. Schefczik and S. Randriamasy, "On prediction of resource consumption of service requests in cloud environments," the 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, 2017, pp. 169-176.

[5] S. Ningning, G. Chao, A. Xingshuo and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," in China Communications, vol. 13, no. 3, pp. 156-164, March 2016.

[6] R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," in IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171-1181, Dec. 2016.

[7] H. W. Kuhn, "The Hungarian method for the assignment problem," NavalRes. Logist., vol. 2, nos. 1-2, pp. 83-97, 1955.

[8] S. Yashpal, S. Alok, "NEURAL NETWORKS IN DATA MINING", Journal of Theoretical and Applied Information Technology, 2005 - 2009 JATIT.

[9] C. Krieger, "Neural Networks in Data Mining", technician report, 1996.

[10] S. Duggal, R. Chhabra , "Learning Systems and Their Applications: Future of Strategic Expert System". Issues in Information Systems, Vol. III, 2002.

[11] S. Chen, R. Lu, J. Zhang, "An Efficient Fog-Assisted Unstable Sensor Detection Scheme with Privacy Preserved", CoRR abs/1711.10190 (2017).

[12] I. Rao and E. Huh, "A probabilistic and adaptive scheduling algorithm using system- generated predictions for inter-grid resource sharing," Journal of Supercomputer, 45, pp: 185-204 (2008).

[13] N. Moustafa, I. Al Ridhawi, and A. Hamza, "An Intelligent Dynamic Replica Selection Model within Grid Systems, " in Proc. 8th IEEE GCC conference on Towards Smart Sustainable Solutions, pp. 1-6, 1-4 February 2015.

[14] R. Buyya, and M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", John Wiley & Sons Ltd, 2002.