Data Caching and Selection in 5G Networks Using F2F Communication

Ismaeel Al Ridhawi[1], Nour Mostafa[1], Yehia Kotb[1], Moayad Aloqaily[2,3], Ibrahim Abualhaol[2,4]
[1]College of Engineering and Technology, American University of the Middle East (AUM), Egaila, Kuwait
{Ismaeel.Al-Ridhawi; Nour.Moustafa, Yehia.Kotb}@aum.edu.kw
[2]Systems and Computer Engineering Department, Carleton University, Ottawa, ON, Canada
[3]Gnowit Inc., Ottawa, ON, Canada, moayad@gnowit.com
[4]Larus Technologies, Ottawa, ON, Canada, ibrahimee@ieee.org

***Abstract*—As an emergent technology the IoT promises to harness the computational and data resources distributed across different remote clouds. Fog computing extends cloud computing by bringing the network and cloud resources closer to the network edge. As the number of resources contributing to the cloud/fog system grows, so the problems associated with efficient and effective resource selection and allocation. In this paper, we introduce a fog-to-fog (F2F) data caching and selection method, which allows IoT devices to retrieve data in a faster and more efficient way. The proposed solution is based on a data caching and selection strategy using a multi-agent cooperation framework. Caching is achieved by decomposing cloud data into a set of files and then placed into fog storage sites. The selection process is based on a run-time file location prediction technique, which collects and maintains a repository of fog data in the form of log files. When data needs to be retrieved, prediction is made with the aid of these logs and previous successful search queries resulting in realistic run-time location estimates as well as best fog selection. Simulation results showcase the reduced data retrieval latency that enable tactile Internet in 5G. Additionally, results show increased successful file hit ratio leading to a reduced number of repeated downloads.**

***Index Terms*—5G, big data, cloud, fog, F2C, F2F, e2e delay, workflow-net.**

I. Introduction

Increasing demand for data access coupled with limitations in current mobile networks has led to the emergence of the fifth-generation (5G) network. With the rise in the number of mobile devices and service applications, data has grown exponentially over recent years [1]. Cloud has played a dominant role in providing both data storage and computational capabilities. Fog computing came to mitigate the shortcomings of the cloud computing scheme within the Internet of Things (IoT) environment by bringing network, processing, and storage resources closer to the devices. This allows IoT devices to both meet hard-constraints and offload much of its data to the fog. No doubt, that 5G combines both cloud and fog computing to accommodate for the anticipated explosive growth of mobile users' data traffic.

Traditionally, IoT devices access data through a remote cloud data storage site. This incurs high delays and network bandwidth overload. The fog computing paradigm was introduced to solve issues related to data access and processing. IoT devices can both access and send data to the fog for local data access and processing. Fog computing extends the traditional cloud computing paradigm by bringing the cloud resources such as storage sites closer to the network edge. This allows for a substantial number of requests to be processed near the IoT devices, thus reducing communication delay and providing faster service [2].

A novel approach in fog computing called Fog-to-Fog (F2F) communication was introduced earlier to determine the best fog to process and/or store a particular task [3]. Fogs are not limited to either execute a task or forward it to the cloud but also have the capability to collaborate with other neighboring fogs. This will reduce data access time and minimize the overall end-to-end (e2e) latency. In this paper, the concept of fog computing and F2F communication is extended further to provide fast data access to IoT devices through data replication and caching techniques at fog storage sites. Cloud data is decomposed into a set of files that if decomposed any less will not add any more value to the decomposition. The decomposed set is then cached into fog storage sites. When a data request is submitted, the query is first assessed by a set of agents to decide whether the request can be answered by the cache.

Additionally, a multi-agent cooperation framework is proposed to achieve the required task of data retrieval and caching. If the requested data (or part of it) is not available in the fog serving the concerned IoT device, then data is retrieved from other nearby fogs, if available, using a run-time file location prediction technique that relies on users' historical executions. The solution considers certain parameters such as the user ID, filename, and resource ID to predict file locations

978-1-5386-3531-5/17/$31.00 ©2017 IEEE

required for new jobs. The proposed approach uses a search technique that relies on users' past history to predict a file's location.

The remainder of the paper is organized as follows. Related work is presented in Section II. Section III covers the proposed solution along with the overall system architecture. The decomposition and caching processes are discussed in Section IV. Section V illustrates the cache selection strategy. Section VI illustrates and defines the proposed multi-agent framework and the workflow model. Simulation results are presented and discussed in Section VII. Finally, we conclude the paper in Section VIII.

II. Related Work

Fog computing extends cloud computing to the network edge, allowing for load balancing, reduced latency, and flexible mobility, which provides a promising solution for 5G networks. Although research is still premature in this area, some authors are now focusing on data replication and caching within fog systems. Kitanov et al. [4] considered evaluating fog computing service orchestration as a support mechanism for 5G networks in terms of round-trip latency. Results demonstrate that 5G will have a great benefit in using the fog/cloud computing environment, where round-trip time is significantly reduced. This will allow 5G to cope with services that require reduced latency, high mobility, and real-time execution.

In [6], the authors proposed a cooperative scheme between nearby fogs to improve QoS of the edge computing infrastructure. Each fog data center uses a buffer to store service requests for future local executions. When the buffer is full, the upcoming requests migrate to a neighboring fog. The neighboring fog will accept to serve the request if its current queue length is below a given threshold.

In [7], the authors proposed a QoS-aware service distribution strategy in Fog-to-Cloud (F2C) scenarios [8]. The work aims at achieving low delay on service allocation by using service atomization in which services are decomposed into distinct sub-services called *atomic services* tailored to enable parallel execution. These atomic services are executed on constrained edge devices. Tasks with higher requirements are allocated on more robust resources and executed in parallel with the atomic services. A control plane within the F2C architecture exists that is responsible for the distribution of the atomic services among the available edge nodes. The authors model the service allocation problem as a multi-dimensional knapsack problem (MKP) [8].

Verma et al. [9] proposed a load balancing method for fog/cloud systems which uses a data replication technique for maintaining data in fog storage sites. The solution aims at reducing the overall dependency on big data centers. The authors in [10] focused on improving users' QoE through load balancing in fog computing. The work considered the case of multiple users requiring computation offloading, where all requests are to be processed by local computation cluster resources. The solution considers a low complexity small cell cluster establishment and resource management customizable algorithm for fog clustering. Simulation results show that the proposed algorithm yields high user satisfaction rates for up to four users per fog with moderate power consumption and high latency gain.

Although there exists recent research in the area of fog/cloud resource sharing and cooperation for load balancing, work in data replication and caching in fog computing systems is still premature. We believe that our work is the first to consider a F2F communication scheme that allows data to be cached within fog storage sites [11]. Fogs then collaborate and share data to complete job requests initiated from 5G network users to reduce the overall latency.

III. Proposed Architecture

As the number of next-generation mobile networks increase, and with enormous amounts of data residing in the cloud, frequently accessed data must be made available in closer proximity to cloud service clients. This will allow for both increased data accessibility rates and enhanced QoS levels. A data caching and selection technique has been developed to overcome data accessibility issues for cloud bigdata in 5G networks. Frequently accessed data is cached on fog storage sites with the aid of the *Data Replication/Caching Module*, where file or block replicas are stored or cached on fog storage sites. Replicas and cached data are regularly updated through notifications sent from the cloud. A data decomposition and caching technique (Section IV) is used to decompose files into blocks. Once data is available in the cache, a selection technique is used to access files or blocks from fog storage sites (Section V). Figure 1 provides an overview of the proposed architecture.

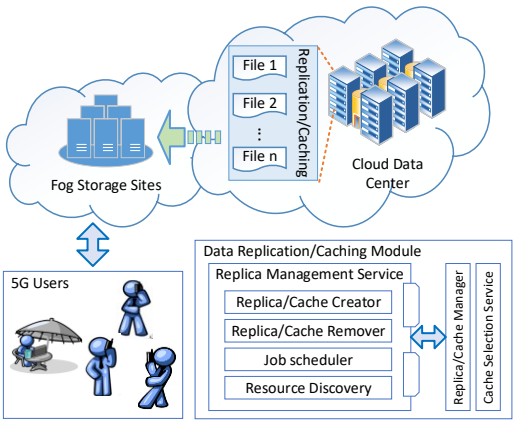

Fig.1. Proposed data replication and caching module incorporated for fog/cloud environments.

IV. Data Decomposition and Caching

We define a *block* to be a set of files that if decomposed any less will not add any more value to the decomposition. A block is an answer to a single file query from a single data set. Any more complex queries can be answered by mathematically composing those blocks together. By increasing the granularity of the cache, a bigger diversity of queries can be answered and the space needed to store information to answer a set of queries is minimized [12].

When a data request is submitted to the fog, blocks are mathematically checked to see whether the fog can fulfill the request. This is achieved by decomposing the submitted request into its own blocks and then comparing those blocks with the cached one. Suppose an IoT device is requesting access to file F_1 = .mp4 media file with English subtitles and special color filtrations. Three blocks exist for F_1: β_1 =.mp4 media file, β_2 =English subtitles for the file, and β_3 =color filtrations for media files. Therefore, when data is replicated from the original cloud storage site to fog storage sites, blocks from a single file are replicated separately either to the same fog or different fogs. Hence, when IoT devices request access to data, the request can be either fully answered, partially answered, or cannot be answered at all.

Assume that a user is requesting access to F_1 and only β_1 and β_2 exist in the fog serving the user (assume Fog_1) as illustrated in Figure 2. Now to complete the user's request Fog_1 will query nearby fogs for the missing block (i.e. β_3). Once the block is found (assume in Fog_2), it is replicated from Fog_2 to Fog_1. This completes the missing data and the returned missing blocks are added to the cache.

To avoid replicating the entire set of data from the original cloud storage site to a fog storage site, block resizing is considered. Block resizing is the process of increasing the cache when a query results in a bigger set of data than is already cached and decreasing the cache when a query results in a smaller set of data than is cached. To resize blocks, hit ratio is considered where blocks that are not queried more than a certain threshold are flushed to assure that the size of the cache is maintainable.

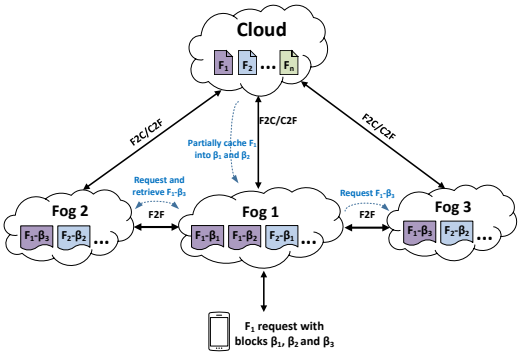

Fig.2. An example illustrating file blocks cached in multiple fogs.

V. Cache Selection

Once file block replicas have been cached in fog storage sites, the process of locating files for job requests must be considered. The majority of distributed systems locate files by using a central replica location repository [6] which maintains indices that represent the mapping between logical cached files and the original files at the cloud. These approaches use a simple match-making approach based on the filename. Replica location repositories work tolerably well for small systems. However, in more complex configurations, such as cloud systems, job turnaround time increases each time a requested file proves not to be registered in a particular replica location repository and an alternative repository has to be sought. Additionally, retrieval strategies must cope with requests for file blocks stored in dispersed fogs.

The proposed file selection model introduced in this paper provides an efficient solution to access local files (i.e. files located in the fog serving the user which requested the file) and remote files (i.e. files located in other fogs). The proposed approach uses a search technique that depends on users' past history to predict a file's location either in local or remote fogs. The proposed solution exploits habitual job parameters from execution logs (user ID, filename, block name, file location, block location, resource ID, etc.) to predict the file/block locations required for new jobs. After a task is completed, the parameter sets which are used to find file locations are stored in a fog file location repository (i.e. edge node). These parameter sets are used to predict file locations for future jobs.

Using the proposed prediction model, the replica management service is able to determine the location of a file in one step and inform the requesting job immediately. Thereby reducing the overhead associated with potentially complicated searches in different fog and cloud storage sites. If a job completes successfully on the basis of file location prediction, the job parameters (i.e. User ID, Resource ID, Required File, Required Block, Block and File Size) are stored in a 'history database' separate from the fog file location repository. This history database is used to support our replica prediction method. Each time a job enters the system, the database is searched, and, if it contains the file location for a particular job configuration (i.e. some instantiation of the job parameters), the result is sent back. Otherwise a new prediction is made, which if successful, is also added to the prediction model for future use. If a prediction is incorrect, or if there is no match in the prediction model for an incoming job, a

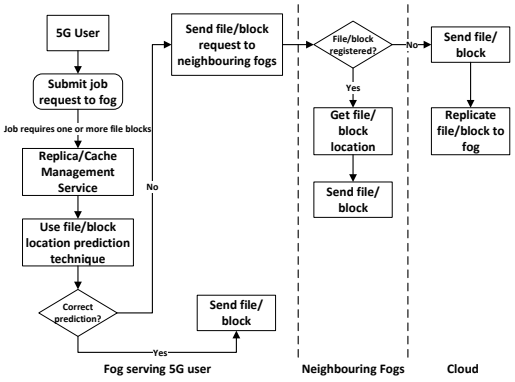

Fig.3. Flowchart illustrating the steps taken to deliver a requested file to a 5G user. If blocks are not cached in the fog serving the user, request will be sent to neighboring fogs.

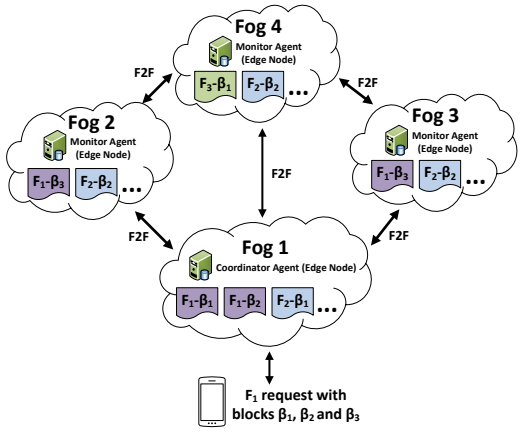

Fig.4. An example illustrating the retrieval of file blocks from neighboring fogs.

conventional cloud storage site search is initiated. Figure 3 provides an overview of the steps taken to retrieve files requested by 5G users using our proposed solution.

VI. Workflow-net Cooperative Fog Model

We assume that the data caching and request problem is modeled using a multi-agent framework. The number of agents involved in the process is greater than or equal to the number of blocks created to cache a particular file. The extra agents involved have the roll of coordinators and monitors for block request submissions to other agents to answer partial queries and collect results from partial queries to build the answer of the whole query. Every time a new block is introduced, a new agent is created to manage the process of dealing with this block. The multi-agent framework is mathematically described as follows:

$$\aleph = \{\alpha, \Psi, \Omega, \tau, \Re, \varphi, \chi\} \quad (1)$$

where

$\aleph$	is the cooperative framework for the agents
α	is the set of cooperating agents in $\aleph$
Ψ	is a set of coordinators and monitors
Ω	is the set of all registered files/blocks in the fog file location repositories
τ	is the set of cached blocks
$\Re$	is a mapping function from agents to blocks
φ	is a mapping function from blocks to repositories
χ	is a mapping function from agents to repositories

and where $\alpha \neq \emptyset$ and $\Psi \neq \emptyset$ and $\alpha \cap \Psi = \Psi$ and $\Omega \neq \emptyset$ and $\|\tau\| \leq \|\Omega\|$ and $\Re = \alpha \times \tau$ and $\forall \tau_i \in \tau$, $\|\Re(\alpha_j, \tau_i)\| \leq 1$ and $\Re(\alpha_j) \cap \Re(\alpha_k) = \emptyset$ and $\varphi = \tau \times \Omega$ and $\forall \Omega_i \in \Omega$ and $\tau_i \in \tau$, $\|\varphi(\tau_j, \Omega_i)\| \leq 1$ and $\varphi(\tau_j) \cap \varphi(\tau_k) = \emptyset$ and $\chi = \alpha \times \Omega$ and $\forall \Omega_i \in \Omega$ and $\alpha_j \in \alpha$, $\|\chi(\alpha_j, \Omega_i)\| \leq 1$ and $\chi(\alpha_j) \cap \chi(\alpha_k) = \emptyset$.

In other words, the set of cooperating agents cannot be empty, the set of coordinator and monitor agents cannot be empty, and the set of coordinator and monitor agents are selected from the set of agents in the framework. Additionally, the set of registered files/blocks in the fog file location repositories cannot be empty. Every block is managed by one and only one agent and every agent manages a single block. Every block in the file location repository is managed by one and only one agent and every agent manages a single block in the file location repository. Every file in the repository has at least one block and the repository and block that are associated together are managed by the same agent.

Figure 4 provides an illustrative example on how blocks are retrieved from different fogs. When a file request is submitted by the user, it is received by one of the coordinating agents, which in turn determine the blocks needed to compose the file. It then sends the request for every block to its responsible agent. That agent will get the request and calculates whether the time needed to transmit the file is within the limit set by the user. If it is, then the block will be transmitted to the coordinator immediately. If not, then the file will be cached within the requesting fog if the block request threshold has been met for future block requests. The coordinator will gather all the blocks and joins them together, producing the requested file and sends it back to the user. In case that the request contains a block that has no agent yet, an agent is created for it.

We use workflow-nets to model the behavior of fog agents. Workflow-nets are an extension to petri-nets, in which the latter is a directed graph with two types of nodes, namely places (circles) and transitions (solid rectangles) [13]. Transitions model events that may occur, while places model pre- or post-conditions for transitions. Arcs connect places to transitions and transitions to places. Workflow-nets are preferred over petri-nets due to their characteristic of having a single source node and a single sink node thus achieving the notion of soundness [14]. Figure 5 shows the flow of

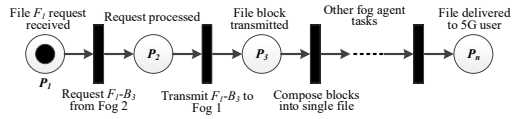

Fig.5. A workflow-net model for an agent controlling a file block request and composition.

behavior for a single fog agent controlling a file block request and composition.

VII. Simulation Results

To simulate the complexity of real cloud systems, a comparison of the proposed caching technique and a full replication method is performed using GridSim [15]. We assume that 5G users reside on geographically distributed sites. The network is modeled as a graph $G = (N, B)$ where the set of nodes $N = \{1, \dots, n\}$ represent storage sites in the fogs and B represents the bandwidth. All nodes are assumed to have uniform bandwidth, computing power, memory and storage capacity. Different scenarios are simulated by varying the number of files, size of files, number of job requests and capacity of storage nodes as outlined in Table I to support realistic large-scale data-dependent systems.

TABLE I. Simulation Scenario Configurations

Number of Fogs	3
Number of 5G Users	150
Storage Nodes per Site	Between 2 and 40
Size of Files	Between 100 GB and 500 TB
Connectivity Bandwidth	Up to 2000 MB/Sec
Size of Workload	Up to 1500 Jobs

Overall simulation results show a decrease in file access delay. Figure 6 shows that the average response time for file/block access is reduced when compared to the non-caching technique where files are replicated to a single fog rather than caching blocks distributively among different fogs. Four different job requests are used to compare the two techniques: small, medium, large and very large file size job requests.

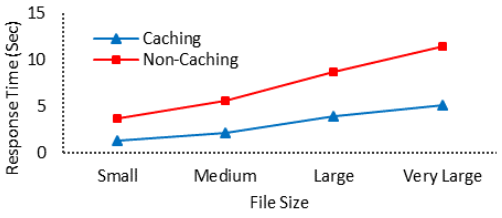

Fig.6. Average response time for file/block access with varying sizes.

The same experiment is repeated while varying the number of job requests. Results depicted in Figure 7 show that access time when using the proposed technique outperforms the non-caching technique in all job request cases.

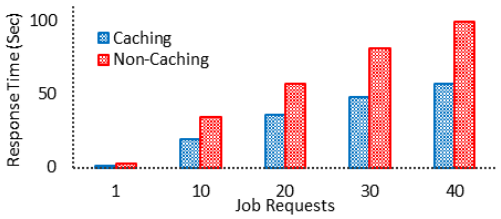

Fig.7. Average response time for file/block access with varying number of job requests.

Additional experiments were conducted to test the job turnaround time (JTT) for both the caching and non-caching techniques. JTT is computed as the average of the total time taken for all jobs to be completed. The system's performance was evaluated under three different scenarios by varying the file size and the number of jobs each time. Results show that the proposed technique's performance outweighs the non-caching solution as the system size increases. Table II provides detailed results of the comparison between the two techniques.

TABLE II. Job Turnaround Time under different scenarios

File size	No. of job requests	JTT for non-caching technique (Seconds)	JTT for caching technique (Seconds)	Average Difference (%)
Small file	500 1000 1500	920 1862 2795	684 1373 2025	26.8
Large file	500 1000 1500	2028 4097 6113	1474 2948 4366	28.2
Very large file	500 1000 1500	5640 11366 17232	3890 7727 11871	31.4
Total	-	52053	36358	-
Average	-	5784	4040	28.8

VIII. Conclusion

Fog computing provides a solution for cloud shortcomings in which network, processing, and storage resources are brought closer to mobile users, hence reducing the overall e2e delay. We introduced a data caching strategy for fog environments in which highly accessed files are decomposed into blocks and cached into different fogs for load balancing. Access to files composed of multiple blocks is achieved by mathematically composing those blocks together. The cache selection strategy relies on a search technique that is trained on users' past history to predict a file's location either in local or remote fogs. The solution exploits habitual job parameters from execution logs to predict the file/block locations required for new jobs. Fog agents are used in the communication process for

block retrieval and composition. Additionally, the problem is modelled as a workflow-net. Simulation results show overall decrease in file access time and job turnaround time when comparing the proposed caching technique to a non-caching method.

REFERENCES

[1] R. Villars, C. Olofson and M. Eastwood, “Big data: What it is and why you should care”, White Paper, IDC, MA, USA, 2011.

[2] M. Aazam and E. N. Huh, "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm," in IEEE Potentials, vol. 35, no. 3, pp. 40-44, May-June 2016.

[3] W. Masri, I. Al Ridhawi, N. Mostafa and P. Pourghomi, "Minimizing delay in IoT systems through collaborative fog-to-fog (F2F) communication," in Proc. 2017 9th International Conference on Ubiquitous and Future Networks (ICUFN), Milan, 2017, pp. 1005-1010.

[4] S. Kitanov and T. Janevski, "State of the art: Fog computing for 5G networks," 2016 24th Telecommunications Forum (TELFOR), Belgrade, 2016, pp. 1-4.

[5] R. Beraldi, A. Mtibaa and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," 2017 2nd International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, 2017, pp. 94-100.

[6] A. Sulistio, U. Cibej , B. Robic, and R. Buyya, “A Toolkit for Modelling and Simulation of Data Grids with Integration of Data Storage, Replication and Analysis”, Elsevier Science, 17 January 2006.

[7] 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.

[8] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan and G. J. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," in IEEE Wireless Communications, vol. 23, no. 5, pp. 120-128, October 2016.

[9] S. Verma, A. K. Yadav, D. Motwani, R. S. Raw and H. K. Singh, "An efficient data replication and load balancing technique for fog computing environment," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 2888-2895.

[10] J. Oueis, E. C. Strinati and S. Barbarossa, "The Fog Balancing: Load Distribution for Small Cell Cloud Computing," 2015 IEEE 81st Vehicular Technology Conference - Spring, Glasgow, 2015, pp. 1-6.

[11] S. Singh, Y. C. Chiu, Y. H. Tsai and J. S. Yang, "Mobile Edge Fog Computing in 5G Era: Architecture and Implementation," 2016 International Computer Symposium (ICS), Chiayi, 2016, pp. 731-735.

[12] M. Khan, and M.N. Khan, “Exploring Query Optimization Techniques in Relational Databases,” in International Journal of Database Theory Application, vol. 6, 2013.

[13] I. Al Ridhawi, Y. Kotb, M. Aloqaily and B. Kantarci, "A probabilistic process learning approach for service composition in cloud networks," in Proc. IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, 2017, pp. 1-6.

[14] Y. Kotb and E. Baderdin, “Synchronization among activities in a workflow using extended workflow petri nets," in Proc. of the 7th IEEE International Conference on E-Commerce Technology, 2005, pp. 548-551.

[15] 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.