2012 Fourth International Conference on Intelligent Networking and Collaborative Systems

Optimizing the energy efficiency of message exchanging for service distribution in interoperable infrastructures

Nik Bessis[1], Stelios Sotiriadis[1], Florin Pop[2], Valentin Cristea[2]
[1]School of Computing & Maths, University of Derby, Derby, United Kingdom
[2]University "Politehnica" of Bucharest, Bucarest, Romania
(n.bessis, s.sotiriadis)@derby.ac.uk (florin.pop, valentin.cristea)@cs.pub.ro

Abstract — **Optimizing the competence of message exchanging algorithm in large-scale grids and inter-clouds could be proven to be a crucial factor in achieving an efficient resource management. Traditional solutions usually incorporate either probabilistic or flooding message exchanging techniques mainly due to the dynamic and unpredictable formation of the infrastructure. Our vision encompasses a total decentralized nodes topology in which message exchanging algorithm allows dissemination of communication messages within a decoupled node formation setting. We consider various e-infrastructure nodes that exchange simple messages with linking nodes regarding resource competence for executing certain service(s) requirements. The optimization criterion is to improve the energy efficiency of the network performance for message exchanging in two cases as follows. Firstly, a requester sends messages to all interconnected nodes and gets messages only from resources available to execute it. Secondly, the requester sends one message for all of the jobs of its local pool and gets a respond from available nodes, and then obtainable resources are ranked and hierarchically categorized based on the performance criterion e.g. latency competency. The algorithm is simulated and results obtained are very supportive.**

Keywords: Message exchanging, grids, inter-clouds, decentralized message passing, energy efficient resource management.

I. INTRODUCTION

This work is motivated by the message passing conceptual design and its algorithms that include a form of communication between different processes and objects in distributed and parallel computing. Our vision is inspired from that model and encompasses a fully decentralized topology of collaborated and decoupled nodes suitable for resource management in large scale and dynamic grids and inter-clouds. In such settings, we aim to explore the message exchanging framework among nodes as happened in message passing algorithms that allows communication among processes of parallel hosts during the actual processing time. This implies that processes among different nodes transfer data (sending and receiving messages) through shared memory areas.

Various algorithms and approaches have been designed with the aim to optimize the efficiency of the message passing algorithmic models. However, most of these works are strongly influenced and driven by congestion parameters of the network topology as well as are related with a homogeneous and tightly coupled setting. In addition, for the case of grid computing, the message-exchanging framework is considered as a generic collective approach wherein all nodes could communicate with each other. Specifically, nodes request for resources by broadcasting messages to any interconnected node; then gather responses and move to the next resource management step (resource allocation etc.) [8]. In general, message passing and message exchanging methods are two different methods as the former is about exchanging messages among processes [10], while the latter is related to message swapping among computing nodes.

Although message passing and message exchanging are considered as two different schemes, nonetheless both share the same fundamental message dissemination algorithms (e.g. All-to-All [2], allgather [1]). Thus, in this work we present an optimization algorithm with regards to the consumed energy efficiency. In this study, energy efficiency is defined as the consumed energy (KWH) and the computational performance of the network for executing detailed demands (average execution times). In advance, our proposed solution is based upon user specified services (jobs or cloudlets). Herein, the proposed solution suggests that by minimizing the number of messages (sent and received) we can achieve an optimization of the job execution performance. For demonstrating the effectiveness of the approach we utilize an inter-cloud setting [6]. In general, the inter-cloud approach expands the cloud capabilities for achieving a wider distribution, yet by retaining global resource utilization equilibrium among various resource pools [5].

In this e-infrastructure we develop a solution for exploring the latency of the system due to message exchanging prior to the execution of services as submitted by their users when random topologies are taking place. The nodes are components of the inter-cloud such as users, brokers, and cloud datacentres that interchange messages during service life-cycle. In addition, we implement our algorithm to achieve a two-fold solution. Firstly, a node sends messages based on the All-to-All algorithm (all nodes send to all interconnected nodes) and responses are collected back only from resources available to execute it [2]. Secondly, a node sends a message for all of the jobs contained within the local pool and gets a response only from available nodes to execute the demand. Lastly available nodes are ranked and hierarchically categorized based on the performance criterion of latencies in service execution times. The rest of the paper is organized as follows. Section II presents the related works in the area of message passing and message exchanging algorithms.

978-0-7695-4808-1/12 $26.00 © 2012 IEEE
DOI 10.1109/iNCoS.2012.16

Section III presents the conceptual design model, section IV the algorithms of the message exchanging setting, section V the scenario cases and section VI the simulation results. Finally, section VII concludes the study with a discussion of the remarks and future research steps.

II. Related Works

Over the years a variety of message exchanging algorithms have been developed in order to address issues depending on the actual topology of the setting. Similar other solutions consider the number of packets to be moved among processors of cluster based parallel and grid computing systems. In such cases the Message Passing Interface (MPI) [13] has been introduced by the academia as a standardized portable message passing system. Different algorithms are related with the MPI basic solutions related with the point-to-point communication and the collective communication approach. The simplest of the cases is the point-to-point in which the processor of a node machine sends a message to another. This includes a diversity of variations in terms of message exchanging properties. For instance, the synchronous case includes the sending of completion information while the asynchronous contains data regarding the time that the message left from the first node. In addition, asynchronous allows high dynamic-ness of the system.

Others include the blocking operation that only allows messages to return after the completion of the process in contrast to the non-blocking operation that tolerates the return of messages straight away [10]. However, this solution is considered appropriate for small scale clusters due to the one-to-one communication pattern. A different approach is the collective communication that involves the routing of numerous processes at a time. This includes operations like the broadcasting (one to many communications). This solution increases the design complexity as it encompasses the synchronization of processes. Nevertheless, it is a more advanced approach which could be applicable for largeer scale distributed infrastructures. In general, by using broadcasting named as "broadcast call" one node sends a message to all nodes of the group. The "reduce call" procedure is called by the MPI at the end for collecting information from all nodes' processors regarding the desired operation and stores the result on one node [2].

Various collective communication procedures include different routines for implementing different behaviors as contained in [13]. This includes the All-to-All, the allgather, the BCast, AllReduce and other functions. As this work is related with message exchanging in large scale infrastructures (grids and inter-clouds) we consider the collective pattern solution as the most appropriate mainly because it involves complete exchange in a one to many model. Initially, the All-to-All model allows complete information among all the node processes of a group. Similarly to the allgather the latter solution collects processes and then broadcasts (BCast) to each conducted node. However, as the scale of the setting increase in number of nodes, the efficiency of the methods performance is influenced significantly due to the overcrowding of network resources. Thus, the reduce functions can minimize the number of messages by decreasing data and broadcasting again.

A number of theoretical models have been further developed in order to avoid the network congestion [1]. For example, the spreading simple algorithm allows a node to send data to node (p+i) where p is the process and i the iteration and, receive data from node (p-i+N) mod N where N is the number of processes [1]. A different approach is the one presented in [3], named as ring/bucket/circular algorithm. Specifically, at each iteration i a process p sends data to a node with an index (p-i+1+N) mod N to the rights neighbor of the list. The recursive doubling algorithm [13] requires less time as the number of total transfers is reduced. The MPI make use of the MPICH [11] to recursively reduce number of messages by utilizing a criterion; when the number of processes is a power of 2 uses recursive doubling for small message sizes. Next, for the rest of the messages (large size) it uses the ring algorithm to achieve message dissemination. However, this solution aim to small-scale cluster based parallel computing systems. Authors in [2] propose that the most of these algorithms have been designed for homogeneous and tightly coupled systems.

In the case of high heterogeneous and de-coupled settings (e.g. grids and inter-clouds) solutions divide network into hierarchies. The MPICH-G2 [12] presents twofold algorithms to gather data up the hierarchy using recursive doubling and then broadcast these data by binomial broadcast (according to a probability factor). Similar the MagPIe presented in [4] includes a three stages algorithm to first allgather data at coordinators, second gather data among coordinators and, third broadcast data by coordinators using again a binomial broadcast. The major drawbacks of these are the fact that follows static network hierarchical schemes in modeling decisions [10]. In addition, data transmission is repeated at coordinators thus keeping bandwidth values in high layers of hierarchy in low levels [1].

In contrast with these approaches the work of [1] illustrates an algorithm that is network topology aware and adaptive to various network loads. This solution follows the transient clustering of nodes based on network characteristics. In a similar vein, [2] focus on an alternative algorithm for minimizing the number of steps through a wide-area network. They also claim that the reduction has a direct impact on the performance modeling by minimizing the factors that directly interfere the wide-area communication. Although efficient algorithms have been developed for specific networks, a generic model for heterogeneous and decoupled nodes has been proven to be complex to be designed. This is mainly because of the dynamic nature of the resources. In advance, an important requirement to be considered is the real-time information processing and the elasticity and scalability of the services (jobs) submitted by the users.

III. Conceptual design of the Message Exchanging Algorithmic Model (Graph Theory)

This section presents the conceptual design of the message exchanging algorithmic model for inter-cloud. To

illustrate this setting we employ graph theory modeling for achieving two-fold benefits. First characterizing internal components into graph vertices (nodes) and secondly showing the interconnectivity structure among pairwise nodes known as edges by means of graph theory theorems. In view of that, let assume that an inter-cloud is a graph $G = (V, E)$ which at an initial stage is an undirected graph with nodes $v_1, v_1 \in V$ and $v_1 \neq v_1$. If v_1 communicates with v_2 there is a trail among nodes called e_1. In our case we aim to a directed graph, thus e_1 is considered as a walk that connects v_1 and v_2 without linkage restrictions (communication can be repeated). Thus, the analysis of the inter-cloud life-cycle will allow us to identify the crucial components and eventually map each of which into graph theory nodes. In advance, we will identify the communication links (edges-walks of the graph).

An inter-cloud is comprised of a set of sub-clouds that is constructed in a fundamental hierarchical form. This includes a data-center that manages physical machines (hosts) and generates a number of virtual machines within a host(s) for sandboxing user demands. Let us assume that a user request for certain requirements contained into a service level agreement (e.g. computational power) using a cloud interface. Then the cloud assigns a transitional component per user to act as intermediate in the communication among user and cloud. This component is named as meta-broker and acts on-behalf of user by ensuring that the cloud does not violate the established service level agreement. For each of the users the cloud system generates a number of meta-brokers by repeating the same process.

When the system extends to an inter-cloud, it is assumed that various clouds could establish communication and grant authority to meta-brokers for message exchanging through inter-relationships inspired by the meta-computing paradigm [8]. Specifically, during the meta-broker development the cloud provides a list of inter-collaborated meta-brokers that have been already initialized at a previous stage. We detail the modeling of this approach in [7]. Next we focus on the mapping of internal components to graph theory initiatives. The next sections present the theorems of the inter-cloud graph theory for mapping components towards the statement of the message exchanging model.

A. Theorem 1. The directed graph formation

Let assume that a user u_1 establishes connection with a cloud c_1 meta-broker mb_1 and sends a request message for a number of services s_1 named as cloudlet. Then mb_1 delegates a list of inter-linked meta-brokers $mbList_1$ forwards this request to an inter-connected meta-broker mb_2 where $mb_2 \in mbList_1$. We assume that at a previous phase a user u_2 executes a number of services in a cloud c_2 meta-broker mb_2. In addition, each meta-broker accesses a local datacenter (dc_1, dc_2) of the cloud for quoting resource availability. Thus, at this stage we have a directed graph G that represents the inter-cloud setting wherein $u_1, u_2, mb_1, mb_2, dc_1, dc_2 \in V$ and $e_1 = (u_1, mb_1)$, $e_2 = (u_2, mb_2), e_3 = (mb_1, dc_1), e_4 = (mb_2, dc_2)$ wherein of $e_1, e_2, e_3, e_4 \in E$ of $G(V, E)$. It should be mentioned that e_1, e_2 are repeated as many times as the number of services submitted by the users. Figure 1demonstrates the graph theory nodes and edges by mapping internal inter-cloud components.

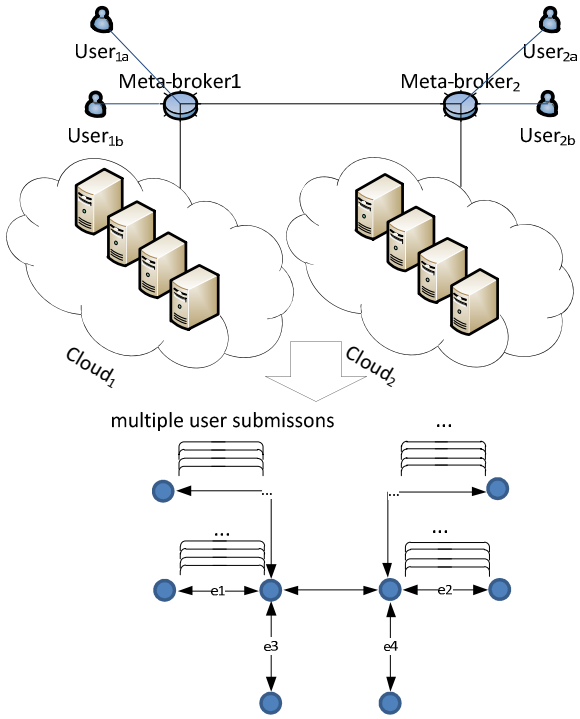

Figure 1: The inter-cloud graph theory mapping model

B. Theorem 2. The sub-graphs formation

In an inter-cloud graph $G = (V, E)$ we define a sub-graph $G_1 = (V_1, E_1)$ and a sub-graph $G_1 = (V_1, E_1)$ to be the sub-clouds of inter-cloud graph. Thus, $V_1 \subseteq V$, $E_1 \subseteq E$ and $V_2 \subseteq V$, $E_2 \subseteq E$ wherein each edge of E_1 and E_2 are incident with vertices in $V_1$and V_2 respectively. Each sub-graph represents a sub-cloud internal graph. That includes multiple users (nodes) that each of which submits multiple services with different requirements. These are directly sent to each user customized meta-broker that again re-forwards a request for availability to interlinked meta-brokers as well as its internal resources (datacenter). Then, the responding nodes receive the requests and decide whether or not their datacenters could execute the job or not by generating a ranking property related with the competence of the datacenter computational power. Specifically, the ranking approach is presented in directed graphs by the Hamiltonian path approach as in [14]. In our solution we assume that meta-brokers collaborate in a Hamiltonian path setting with other meta-brokers and datacenters (each node visits contacted node only ones).

C. Theorem 3. The ranking tournament

Our model incorporate this solution by assuming that a graph G_n^* contains sub-graphs G_1, G_2 wherein $G_1 \subseteq G_n^*$ and $G_2 \subseteq G_n^*$ are directed graphs that contain n vertices. For each vertices $v_1, v_2 \in G_n^*$ there is an edge(e_1, e_2) implied from a ranking tournament of responding node. In such case

a requesting node asks every responding node exactly once by sending and receiving messages, then it ranks the responding nodes by a ranking criterion (e.g. latency of responding time, availability, competence of computational resources, priorities, etc.). The requesting node collects addresses of responding nodes from an internal list. Then it generates paths that are bi-directed walks named as p_i wherein $i \geq 2$ that contains $i-1$ edges e.g. (n_1, n_2), $(n_2, n_3), \ldots (n_{i-1}, n_i)$. A node is part of the list only and only if $i = n$. In any other case n is a vertex that does not appear in p_i, thus it is not a walk.

D. The model of message exchanging

The message exchanging solution maps the graph theory and Hamiltonian path (for meta-brokers) characteristics as follows. We assume a send request message to a meta-broker as presented in Theorem 1. The last one forms the requesting node that forwards resource competence demand to interlinked meta-brokers that are members of sub-clouds as illustrated in Theorem 2. At last, according to Theorem 3, contacted meta-brokers rank resources according to a criterion and send responses back to the requesting node. This case represents the traditional inter-cloud communication pattern for message passing in which nodes create a transient graph according to cloud suggestions for collaboration. In advance, the message passing model is an All-to-All solution if the user submits one job, while it is transformed to allgather approach [1] when a user submits more than one request. Thus this forms the energy efficiency message exchanging algorithmic statement.

A requesting node (user) sends a message that contains a requirement specification to a responding node (contacted meta-broker) that contains all the cloudlets require to be executed. The last one communicates with interlinked meta-brokers by sending a message that encompasses the list of jobs. The contacted meta-brokers rank internal resources for each received job by executing a performance criterion function concerning the latency of components (meta-brokers, datacenters) on replying back. Then they send a message that contains only the jobs that are capable of executing enclosed in the initial list. The requesting node collects acknowledgements from responders and classifies nodes by ranking each of which through a tournament operation. Finally, the meta-broker moves to the resource allocation step.

This model offers significant advantages over the traditional solutions presented in section 2 as incorporates dynamic management features as follows:

a) It supports a diversity of variations in terms of message exchanging properties (e.g. performance criterion).

b) It allows synchronous meta-brokering inter-linking due a user request.

c) Collectively distributes the whole set of jobs instead of each separately.

d) It gets responses only from high ranking resources per job requirement.

e) It offers high adaptive-ness to various workloads as request resource availability during run-time.

f) It supports message exchanging of the inter-cloud topology (topology awareness).

g) It deals with user specified services, thus service-orientation message exchanging.

h) It executes real-time information retrieval as requests for availability happen during the message exchange life-cycle.

i) It offers fully decentralization (for meta-brokers) and elasticity due to the variety of user defined tasks as well as heterogeneity by sandboxing tasks in virtual machines

Relevant algorithm pseudo-codes are discussed next.

IV. THE PROPOSED ALGORITHMS

This section presents the algorithmic model of the message exchanging approach between components (nodes) of the inter-cloud. This includes a two-fold solution that incorporates the request and response case for users and meta-brokers respectively. In addition, we present the performance criterion for ranking resources (according to theorem 3) which is the latency of the brokers and datacenters on replying back. Thus, we focus onto the message exchanging procedure that exists prior to the resource allocation process that currently is out of the scope of this study.

A. The User Message Request Algorithm

The user requests for resources by sending a message with the cloudlet specification (e.g. job length, processor number etc.) for each of all the jobs. This takes the form of an array list ($cloudlet[i, spec]$) where i is the identification number of the cloudlet and spec is the specification of the cloudlet). Algorithm I shows the procedure which accepts the user request as an input and waits for the meta-broker to function for a specific amount of time.

Algorithm I: User Message Request

```
1. function userRequest(message)
2. input: userCloudlet_i
3: for each userCloudlet_i ∈ userList_i
4:         create cloudlet_i.spec_i(length, mips, pesNumber)
5:         output: m_i = userCloudletList(cloudlet[i, spec_i])
6: end for
7: send userRequest(message) to meta-broker
8: wait for response(interval x)
9: if interval x not violated then
10:        collect mbrResponse
11:        start cloudlet life-cycle
12: end if
13: end function
```

Specifically, the algorithm creates the cloudlet specification for each service. Then it generates a two-column array list wherein the first column denotes the id of the cloudlet and the second the specification. Finally, the list is sent to the meta-broker(s) and the user waits for response(s) during a definite amount of time (interval x). If the interval is not violated it implies that the meta-broker responds on time and the cloudlet life-cycle starts.

B. The Meta-Broker Message Request-Retrieve Algorithm

The algorithm II presents the meta-broker message request-retrieve pseudo-code. The meta-broker gets as input the user cloudlet list formed in algorithm I. Then for each interlinked meta-broker it sends a message by forwarding the list collected from the user. Also, this is sent to the internal datacenter and responses are collected. After a specific interval the responses from contacted meta-brokers are collected and a ranking procedure is started according to a specific latency criterion as given from formula (1). This is the degree of a vertex of the graph G which represents the number of edges incident to the vertex, with loops counted twice (request and response). We assume that at least one of the cloudlets could be executed ($deg(mbr_i) \neq 0$), consequently the message is counted twice since one message is returning always back.

$$Latency_{mbr_i} = \sum_{mrb_i \in mbr} deg(mbr_i) \quad (1)$$

This is the performance measure or the time elapsed till the response of the node. Specifically, the study will explore this property in the experimental analysis section. Finally, the required resource is set and the algorithm moves to the resource allocation mechanism. The last one is currently out of the scope of the study, thus we assume that this procedure sends the VM address to the user. A detailed discussion on this is illustrated in [15].

Algorithm II: Meta-Broker Message Request-Retrieve

```
1. function mbrRequest(message)
2. input: userCloudletList(cloudlet[i, spec_i])
3: input: mbrResponse(message)
4: set message = userCloudletList(cloudlet[i, spec_i])
5: for each requestmbr_i ∈ mbrList
6:         send mbrRequest(m)
7: end for
8: send mbrRequest(message) to internalDC
9: get internalDC response
10: start trigger for mbrResponse(message)
11: set triggerInterval = 0 and wait for response(interval y)
12: if triggerInterval = y then
13: set flag = TRUE
14: end if
15: if interval y not violated and flag=TRUE then
16:         run ranking(LatencyCriterion, rankTable[i, rank])
17:         set results
18: end if
19: go to resource allocation mechanism
20: output: cloudlet life-cycle
21: send cloudlet life-cycle to user
22: end function
```

C. The Meta-Broker Message Response Algorithm

Algorithm III illustrates the meta-broker message response case. Specifically, the algorithm gets as input the user cloudlet list for the meta-broker and requests for resource availability from internal datacenter. Then, for each cloudlet executes a ranking procedure and places results in a ranking table (a two column array that holds in the first column the id of the job and in the second column the ranking value). It should be noted that the ranking value is considered according to a) the availability of the datacenter to execute rthe equest, b) the latency of the datacenter in replying back and c) a coefficient variable for controlling further simulation characteristics (e.g. reservations). At last a response is send back to the requester only in the case of resource availability.

$$Latency_{dc_i} = (\frac{1}{2} * \sum_{dc_i \in n} deg(dc_i) + \sum_{i \in a}^{a} deg(dc_i)) * c \quad (2)$$

Specifically formula (2) computes the latency of the contacted meta-broker as the sum of the responses sent multiplied by ½. This denotes the half loops as the algorithm sends messages but not all the nodes respond back. The nodes that reply are given by the variable a. Thus the sum of the vertices that respond are summed up and added to the half-walks vertices. This value is again multiplied with a coefficient property c that represents priority jobs or advance reservation mechanisms etc.

Algorithm III: Meta-Broker Message Response

```
1. function mbrResponse(message)
2. input: userCloudletList(meta-broker_i, cloudlet[i, spec_i])
3: send mbrRequest(message) to internalDC
4: get internalDC response
5: for each userCloudlet_i ∈ userList_i
6:         run rankingProcedure(PerformanceCriterion)
7:         create rankTable[i, rank]
8:         output: message = rankTable
9:         for each i∈ rankTable
10:                 send mbrResponse(message)
11:        end for
12: end for
13: end function
```

Finally, the total latency from the perspective of a user is calculated from formula (3) which represents the sum of meta-broker and datacenter latency for the request i (as the set of cloudlets initially submitted).

$$Latency_{user_i} = Latency_{mbr_i} + Latency_{dc_i} \quad (3)$$

To conclude in this section we have presented the algorithmic model as well as the latency functions of the message exchanging system according to the degree of a directed Hamiltonian graph. Next, we discuss the scenario cases for multiple user submissions.

V. SCENARIO CASES

This section presents the scenario cases of the inter-cloud system for demonstrating the message exchanging procedure and the topology awareness and adaptive-ness of our approach. Specifically, we demonstrate two cases of the inter-cloud each of which covers two different topologies as follows. The first setting encompasses a small number of users that submit various workloads to the inter-cloud system. In this case few meta-brokers collaborate for exchanging messages. The second case includes a large number of users that submit one service, so the setting contains a complex meta-brokering topology.

A. Multiple cloudlet submissions

Firstly, we demonstrate a graph theory diagram of multiple cloudlets submitted by two users. In this case, the cloudlets are clustered in a list from the meta-broker mbr_1 and a dissemination message by sending to other meta-broker (e.g. mbr_2). Figure 2 illustrates the multiple cloudlets submission phase. This is a realistic scenario for business and enterprise clients that require executing multiple VMs in a private company's cloud system (e.g. a company that leases VMs for its employees to amplify their everyday activities). It should be mentioned that usually this configuration comprises identical VMs.

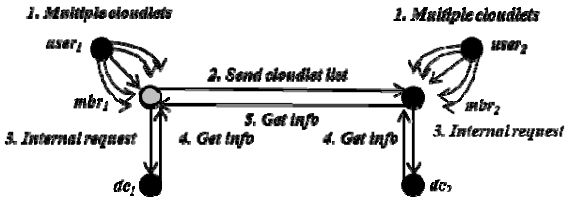

Figure 2: The multiple cloudlets submission phase.

In this topology, the following messages are exchanged in this specific topology.

1. Multiple cloudlets submitted by user1 to mbr1 for resource execution trough a request message.

2. Meta-broker mrb1 forms a list with cloudlets and forwards a message referencing the list in mrb2.

3. Meta-brokers send message for resource availability to internal components.

4. Requests collected back from meta-brokers mbr1, mbr2 that both ranks resources.

5. Meta-broker mbr2 sends a message with the list of possible accepted jobs along with the ranking for each job.

6. At last, the requested meta-broker ranks resources, according to the latency functions illustrates in previous section. Finally the procedure moves to the next step (resource allocation).

B. Multiple users submission

The first case demonstrates a graph theory diagram for an inter-cloud that accepts few cloudlets (one per users for our case); however the number of the users is getting large. Figure 3 shows such case wherein four users submit service requests to personalized meta-brokers. The meta-brokers are interlinked in a Hamiltonian path [14] denoting that each meta-broker visits each other only once. Messages for resource availability are exchanged among internal hosts and meta-brokers that finally rank performance (per cloudlet) and re-forward results back to requester. Similarly, the same steps are followed as happens in case 1. This is a realistic scenario for everyday cloud clients that require executing one or few VMs in a cloud system by leasing computational power (e.g. an Internet user that lets an amazon VM client). Usually, such clients have fully customization power thus they request a great diversity of resources.

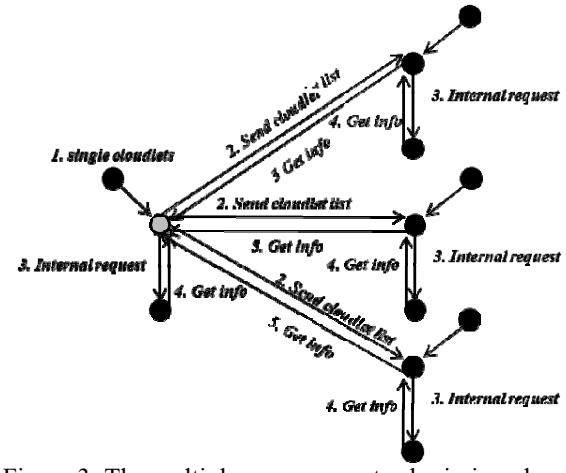

Figure 3: The multiple users request submission phase.

This section demonstrated the elasticity of the model in terms of handling different user demands. Moreover, the approach offers a collective, topology-aware and real-time processing information solution. The next section presetns the experiment and configuration setup along with a discussion of the usefulness of our approach in terms of simulating the aforementioned topologies.

VI. SIMULATION AND RESULTS

This section presents the basic experimental analysis for demonstrating the effectiveness of the message exchanging algorithm. We integrate our solution in the CloudSim [9] which is a framework for modeling cloud infrastructures and services. In this setting we implement an inter-cloud to simulate scenarios of figure 2 and 3. By developing such networks we focus on measuring the average execution times of the directed graph for different configurations (number of users, brokers, and cloudlets) and latency variations. In addition, we measure the energy competency in terms of energy consumption. Next we illustrate the variation of computational performance for the simulation of a) a graph theory model and the performance of multiple cloudlets (five per user) submitted by two users with different configurations and b) a graph theory model for an inter-cloud that accepts few cloudlets (one per user for our case); however the number of the users is increased to four.

The benchmark for comparison is considered to be the traditional message exchanging model of broadcasting requests and get responds for each cloudlet individually. Thus we measure the latency that represents the message walks. The first experimental setting contains the configuration according to topology of figure 2. We run two cases each of which contains two configurations for cloudlets requirements. These include a user that submits five cloudlets in a meta-broker. For each case we test multiple latency values with regards to the traditional message exchanging model and the proposed solution. Figure 4 illustrates the weights w_1, w_2, w_3 that are simulating. Specifically, we assign a value to each of these for calculating the total walks (formulas of section IV)

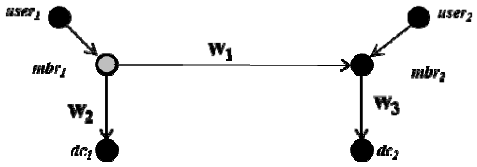

Figure 4: The topology of the directed graph for case 1.

Figure 5 presents the average execution times of the inter-cloud for a variation of latency times (10 to 200, increased by 5) where cases 1 (C1) represents the traditional model and case 2 (C2) our solution. We run both cases using the same cloudlet configuration.

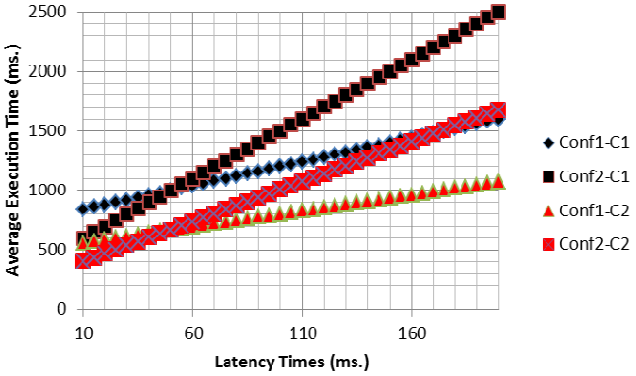

Figure 5: The average execution time per latency variation of case 1 and case 2 for 2 users' submissions

It is apparent that the C2 outperforms C1 as the actual execution times increased considerably even for different user configurations (C1 cloudlet1 length is $8*10^4$ while C2 cloudlet2 length is set to $4.5*10^4$). Execution times are affected significantly as the lines (C2) offers optimized performance for both users. In the second case four users submit one cloudlet thus a new topology is generated. Figure 4 illustrates the weights w_1 to w_7 that are simulated.

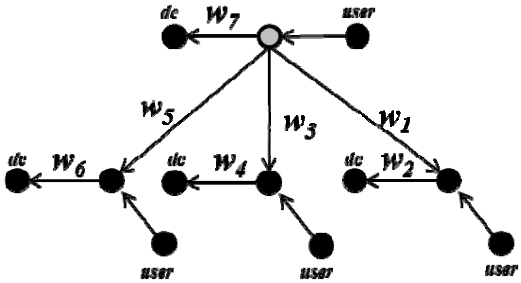

Figure 6: The topology of the directed graph for case 2.

Figure 6 presents the average execution times of the inter-cloud for a variation of latency times (10 to 200, increased by 5) for the four user submissions. We run both cases using the same cloudlet configuration while again we alter the message exchanging algorithm.

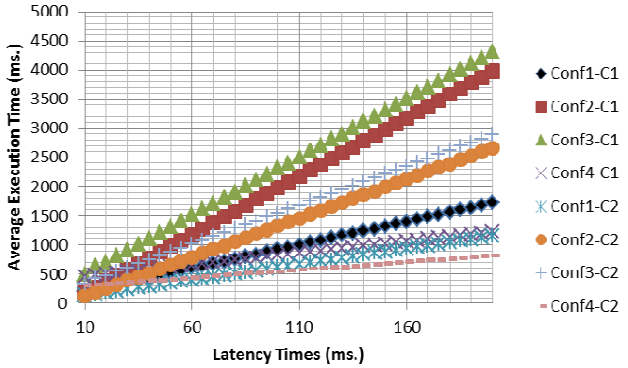

Figure 6: The average execution time per latency variation of case 1 and case 2 for 2 users' submission of one cloudlet.

It is apparent that in this experimental configuration case 2 (C2) outperforms case 1(C1) due to the optimization of the average execution times. The value of cloudlets is set to one thus the average execution time is equal to the actual execution time of the cloudlet. Finally, figure 7 demonstrates the actual energy consumption of the hosts when running the first case configuration for exploring energy efficiency. It should be mentioned that the values of KWs are particularly low due to the small value of the simulation execution times (ms.). In addition the figure demonstrates the linear trend lines for each configuration.

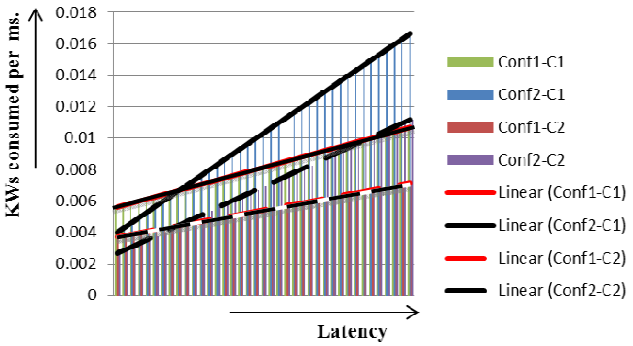

Figure 7: The KWs consumed per ms. for latency times of first experiment along with the linear trend lines

Further to that, the figure 7 shows that the trend line of our solution rises while the trend line of the traditional solution decreases in higher rate. This is an important gain as the energy consumption for high latency indicates a reduction tendency. This is particularly useful for large scale inter-clouds wherein the number of actors (users, meta-brokers, datacenter) is massive, therefore by adopting this model we could achieve an efficient optimization of average execution times and energy consumption measures. It should be noted that case 2 shows identical energy consumption measures. Formula (4) demonstrates the energy consumption function (in KW) with regards to the datacenter host consumed watts; the time elapsed during the cloudlet life-cycle, the cost per KWH (average) and a coefficient value as an experimental property for adjusting simulation results.

$$ConsKW = \frac{watts \times time}{1000} \times costPerKWH \times coef \quad (4)$$

Specifically, the watts are set to 300 (average value of a high-power workstation), the cost for KW is set to 8 cents per hour (UK bases) and the coefficient value is set to 10 to slightly increase the values due to low workload. To conclude, this section presented the inter-cloud message exchanging model when compared with the traditional solution of non-advancing message exchange decisions.

VII. Concluding Remarks and Future Work

The simulation experiments draw the following considerations that meet the posed requirements of the design issues as presented in section III.

a) The diversity of message exchanging latencies shows increased performance (execution time, energy consumption).

b) The collective model (operating in synchronous standards) improves the computational performance (e.g. for the first experimental case the improvement factor is 1.4)

c) Ranking procedure is considered as first come first served fashion, and for this case the energy consumption levels are improved.

d) Both experimental cases show high adaptive-ness to various workloads and topologies as well as user-orientation service provisioning.

e) The real-time processing of information it affects performance (due to real-time exchanging of messages) however it offers a realistic solution (non-static).

f) The decentralization offers high dynamic-ness by slightly affecting performance due to meta-brokering message exchanging delays.

It should be mentioned that a complete knowledge solution is not realistic for large scale settings, thus we assume that one meta-broker per cloud communicates with another meta-broker per inter-connected cloud. Thus, we represent a complete knowledge model in terms of inter-connectivity of sub-clouds. We further detail the design of this model in [6], [7]. In addition, the model presented herein could be utilized for efficient resource management of relevant large scale systems (e.g. grids). The future work includes the exploration of different ranking techniques for achieving a further optimization of our approach.

In addition, we aim of implementing a message passing interface system for queuing host processors for information processing during run-time; thus achieving a more realistic solution. In addition different variations of VMs (number and configuration) could be included to demonstrate the heterogeneity of the system. With regards to energy consumption, measurements required to be validated in various workloads and different topologies for identifying supplementary optimization criteria. At last, during simulation we have utilized simple scheduling algorithms (for VMs and cloudlets) thus a more advanced solution could further improve model results.

References

[1] Gupta, R. and Vadhiyar., S. S. (2007). An efficient MPI_allgather for grids. In Proceedings of the 16th international symposium on High performance distributed computing (HPDC '07). ACM, New York, NY, USA, pp.169-178.

[2] Steffenel, A. L. and Jeannot, E. (2007). Total Exchange Performance Prediction on Grid Environments: modelling and algorithmic issues, Towards Next Generation Grids, Springer US, pp. 131-14.

[3] Chan, E, Van de Geijn, R., Gropp, W.,and Thakur, R. (2006). Collective Communication on Architectures that Support Simultaneous Communication over Multiple Links. In PPoPP '06: Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming, pp. 2–11.

[4] Kielmann, T., Bal, H., Gorlatch, S., Verstoep, K., and Hofman, R. (2001). Network Performance-aware Collective Communication for Clustered Wide-area Systems. Parallel Computing, 27(11): pp.1431–1456

[5] Bessis, N., Sotiriadis, S., Cristea, V., Pop, F. (2011). Towards inter-cloud schedulers: Modelling Requirements for Enabling Meta-Scheduling in Inter-Clouds and Inter-Enterprises, Third International Conference on Intelligent Networking and Collaborative Systems (INCOS 2011) , Nov 30 - Dec 2 2011, Fukuoka, Japan.

[6] Bessis, N., Sotiriadis, S., Xhafa, F., Cristea, V., Pop, F. (2012). Meta-scheduling issues in interoperable HPCs, Grids and Clouds, International Journal of Web and Grid Services, volume 8, no 2.

[7] Sotiriadis, S., Bessis, N. and Antonopoulos, N. (2012). Decentralized Meta-brokers for Inter-Cloud: Modeling Brokering Coordinators for Interoperable Resource Management, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'12), May 29-31, Chongqing, May 29 – 31 2012, ISBN 978-1-4673-0024-7/10, p.p.: 2475-2481.

[8] Sotiriadis, S., Bessis, N., Xhafa, F., and Antonopoulos, N. (2012). From Meta-computing to Interoperable Infrastructures: A Review of Meta-schedulers for HPC, Grid and Cloud. In *Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications* (AINA '12). IEEE Computer Society, Washington, DC, USA, 874-883.

[9] Calheiros, R., N., Ranjan, R., Beloglazov, A., De Rose, C., A., F., and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.* 41, 1 (January 2011), 23-50.

[10] Tseng, C. H., Wang, S., Ko, C., and Levitt. K., (2006). DEMEM: distributed evidence-driven message exchange intrusion detection model for MANET. In *Proceedings of the 9th international conference on Recent Advances in Intrusion Detection* (RAID'06), Diego Zamboni and Christopher Kruegel (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 249-271.

[11] Mpich2 home page. Available at: http://www-unix.mcs.anl.gov/mpi/mpich2. Accessed 20/06/2012

[12] MPICH-G2. Available at: http://www3.niu.edu/mpi, Accessed 20/06/2012

[13] The message passing interface (MPI) standard, Available at http://www.mcs.anl.gov/research/projects/mpi/, Accessed 20/06/2012

[14] Bondy, J.A.; Murty, U.S.R. (2008), *Graph Theory*, Springer.

[15] Sotiriadis, S., Bessis, N., and Antonopoulos, A. (2012). Advancing inter-cloud resource discovery based on past service experiences of transient resource clustering, The 3-rd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT-2012) {to appear}