2012 Third International Conference on Emerging Intelligent Data and Web Technologies

Advancing inter-cloud resource discovery based on past service experiences of transient resource clustering

Stelios Sotiriadis[1], Nik Bessis[1], Pierre Kuonen[2]
[1]School of Computing & Maths, University of Derby, Derby, United Kingdom
[2]Department of Information and Communication Technologies,
University of Applied Sciences of Western Switzerland (Fribourg), Switzerland
[1] (s.sotiriadis, n.bessis)@derby.ac.uk, [2] pierre.kuonen@hefr.ch

Abstract — **Resource discovery in large-scale distributed systems is a challenging issue, mainly due to the traditional centralized topologies in which the nodes of these systems are typically organized. This unified approach has been proven to be effectual for cluster-based grids and clouds but questionable for large scale, heterogeneous and dynamically interoperable e-infrastructure such as grids and inter-clouds. The latter e-infrastructures are less tolerant with regards to scalability, elasticity and flexibility. In this work, we explore the decentralized distributed search protocols that transform nodes of large size distributed systems to act as both clients and servers. Specifically, we extend the use of a job profile specification that is generated during the user job submission process. The exploitation of the job profile will allow us to orchestrate and group different user submissions into transient inter-cloud brokering groups that represent a temporary resource cluster according to the current service submission characteristics. We suggest the clustering submissions in an inter-cloud system with the view of forming notional short-term grouped nodes that may reform over the time. In this way, the resource discovery process is becoming dynamically; that is to say, a transient group of nodes is required advancing a single request based on current time; opposed to multiple requests as currently happens. We further propose an architecture that implies nodes orchestration based on previous resource requests as well as we model a service meta-registry for inter-cloud systems to store relevant information to service submission.**

Keywords: Meta-registry, Grids, Clouds, Inter-cloud, Meta-Brokers, Resource Discovery

I. INTRODUCTION

Over the recent years new technologies like computing grids, clouds and inter-clouds are organized in e-infrastructures with the aim of offering substantial capabilities of high computational efficiency for utilization of remote resources. These settings contain computing machines (nodes or peer) that are relying in a remote location and could be utilized by users in the form of on-demand offered services. Massive computing capacity (hardware and software), resides in a grid Virtual Organization (VO) or a cloud datacentre and could be delivered in a variety of service formats (software, hardware, infrastructure as a service). In a generic view, these are identical to job submissions that have been encapsulated in application execution requests and are about to be executed by a computing CPU. In our work we utilize this model in the inter-cloud system which is an extension of the Internet that contains decoupled computing sub-clouds that exchange services. Eventually, the services will be sandboxed and executed in a virtual machine (VM) of a cloud datacentre host.

However, our setting aims to an interoperable cloud partnership. The need for that is relying on the fact that cloud computing evolves rapidly due to the increase number of its market value, thus the quantity of the users increases as well, though in a non-analogous way. To this extend, we presented the inter-cloud [1] in order to transforms the infrastructure. This lets the setting to go beyond of its premises by allowing cloud datacentres to communicate for service exchanging. In addition, by using virtualization technology VMs could be distributed among various collaborated clouds with the aim of increasing scalability while at the same time reducing the heterogeneity of the system.

In an inter-cloud setting the overall resource management process is a complex decision due to the large number of users and resources [3]. This includes various phases such as resource discovery, availability, allocation and execution. In this work we focus on the resource discovery of inter-clouds by considering an inclusive vision as presented in [2], [8]. Specifically, meta-brokering functionalities incorporated in cloud datacentres in order to achieve a fully decentralized and decoupled setting. This includes the decouple-ness of users and cloud providers for offering a wider service dissemination orchestration.

In such setting, a user submits a request for service allocation in the form of a job specification known as cloudlet. Each cloudlet contains a variety of required physical resources (required CPU, memory, storage bandwidth etc.). This specification is submitted to a cloud representative named as broker. The last one characterizes the role of a node for discovering inter-connected meta-brokers that exist within its meta-registry. Eventually, multiple service requests are distributed in such system for discovering available resources. Our concept includes the transient clustering of inter-cloud local and meta-brokers and their resources according to the service specification. This means that meta-brokers act as decentralized nodes and grouped and reformed over the time to handle different requests notionally during run-time in a dynamic mean.

978-0-7695-4734-3/12 $26.00 © 2012 IEEE
DOI 10.1109/EIDWT.2012.16

Thus, section II illustrates the related works of nodes clustering in large scale systems and section III presents the inter-cloud meta-model that extends the work of [9] with a focus on the service discovery process. The remainder of the paper is organised as follows. Section IV, presents the meta-registry profile discussion by analysing the cloudlet key characteristics. Section V discusses the resource discovery architecture and section VI the modelling of resource discovery algorithm. Section VII illustrates a simulation scenario case for demonstrating the effectiveness of our model. Finally, the study concludes with a discussion of the future research steps.

II. Related Works

Over the recent years, the resource discovery in large-scale systems became a hot research topic by including the orchestration of nodes in various topologies, e.g. centralized, hierarchical and decentralized models. Fundamentally, centralized and hierarchical solutions offer efficient resource discovery times by reducing request disseminations and number of responses. However, when the system extends to a large scale, thus multiple nodes connected randomly, these schemes reduce performance due to the single point of failure and bottleneck of the centralized nodes. In inter-clouds, the increasing diversity of service requests adds important issues that require to be efficiently handled such as heterogeneity, elasticity, large scale distribution, and run-time information processing. Such requirements are very important to and are addressed by the inter-cloud e.g. by sandboxing services in a single VM could overcome heterogeneity. Other issues, like elasticity, scalability and run-time information processing are considered as system dynamics and are addressed by the fully decentralization of the setting. In any case the resource discovery contains resource matchmaking and availability procedures. In our model we propose a decentralized resource discovery model that organizes services and nodes according to the submission profile known as meta-registry. Herein we present a discussion of related works in the area of resource discovery based on nodes clustering criteria.

The work in [4] presents a resource discovery scheme for Grid systems. Specifically, the decentralized resource mechanism aims of overcoming centralized deficiencies by utilizing the concept of resource clustering in a peer to peer network. That is to say a grid node with the same characteristics (resource types) and performance could clustered with similar nodes (according to specific characteristics e.g. identical service submissions) in order to increase scalability of the resource management while decreasing the searching depth. This is achieved by utilizing the P2P hybrid resource discovery framework in order to organise resources into a peer to peer overlay network that cluster similar resources. However, this scheme describes a hierarchical resource discovery mechanism which is not fully decentralized. The fundamental design is based upon the idea of inter-connected peer to peer networks in which some nodes (peers) act as intermediate stations for achieving communication. In addition, our vision incorporates the fact that various nodes might belong to different communities (Virtual Organisations), thus there is no need for inter-connection and eventually creation of bottleneck points in hierarchical connected overlay networks.

The work in [5] discusses a hierarchical grid computing environment of intra-clusters (small-world) clusters. Similar to [4], authors in [5] present their clustering grid work based on overlay network topologies, however by utilizing graph theory for demonstrating the small-world property. The actual characteristic of the nodes clustering is the latency value of the delay in nodes response. Nodes with small latency are collected into a cluster forming a small-world framework. The great advantage of this is the decentralization and scalability of the mechanism which avoids bottleneck on centralized topologies. The authors of this scheme claim that simulation experiments have high success rate, thus resource discovery based on clustering nodes utilizing latency offers efficient performance. However, this solution does not include different job requirements as it is solely refers to the weight among networking nodes (transfer time among nodes). In our perspective, grid computing nodes require to form groups based upon various requirements (i.e. job characteristics for specific resources).

Similar works are the efforts of [6] where authors aim to improve a small-world clustering efficiency by developing routing tables through the use of certain probabilities with respect to diverse properties. However [4] discuss that clusters nodes do not work for resource discovery but according to the principle of graded diffusion from a global perspective. The work in [7] introduces a solution for resource discovery in computational grids by organising nodes dynamically. The main concept is based on the idea that every node require to contribute computational resources to the whole computing grid. This is achieved by designing overlay graphs to have an average low degree (low graph diameter). However, in this work nodes are classified to consumers and produces depending on the job consumption rate in a non-transient way. At last, work in [11] presents a grid resource discovery algorithm that discovers the appropriate resource for a specific request and then effectively directs the request to the resource within that environment. However, this solution is based on matchmaking routers in grid systems.

In contrast to the aforementioned works, our solution cluster resources for inter-cloud systems based on the service specification request. While most of other works group their nodes according to weighted parameters (i.e. distance of peers, or transfer times) and without considering the different job requirements and short-term binding of resources we first group service (job) submissions and secondly nodes (meta-brokers) for binding services in their local resources. Additionally, we aim to a realistic solution that organizes temporary clusters during service submission run-time. That is to say meta-brokers are gathered together when the service submission starts and re-formed at the next service offer. In this context, the next sections illustrate the inter-cloud model that facilitates the meta-brokers and the meta-registry component. Next we discuss the overall decentralized meta-computing operation prior to the service submission phase.

III. THE INTER-CLOUD RESOURCE DISCOVERY SYSTEM

Our design for the inter-cloud system incorporates the utilization of a novel meta-brokering component that conceptually is based upon the meta-computing paradigm. This implies the generation of transient meta-brokers by a cloud datacentre that are unique for every request and discovery service submission. For each user the datacentre binds a personalised broker (local-broker) to control job specification and a meta-broker to control interconective-ness with other meta-brokers belonging to interoperable clouds . Thus, the meta-broker provides an autonomous orchestrator that characterizes the initial point of the cloud from the view of a user. In addition, each meta-broker can access the actual cloud infrastructure (e.g. datacentre characteristics, Hosts, VMs) for resource utilization as the expectation is that during job submission the meta-broker communicates with a local cloud broker for information exchanging.

The local-broker represents the user within the cloud system that interacts with the datacenter for acquiring resources. The main functionality of this component is to hide the low level operations of the cloud from the requester e.g. the creation and destruction of VMs that encapsulate the submitted cloudlet(s). Further, the local-broker communicates with the meta-broker that monitors the whole service submission and execution phases by storing various cloud components. It should be mentioned that the datacenter generates individual personalized local-brokers (along with meta-brokers) for each of the users that access the cloud.

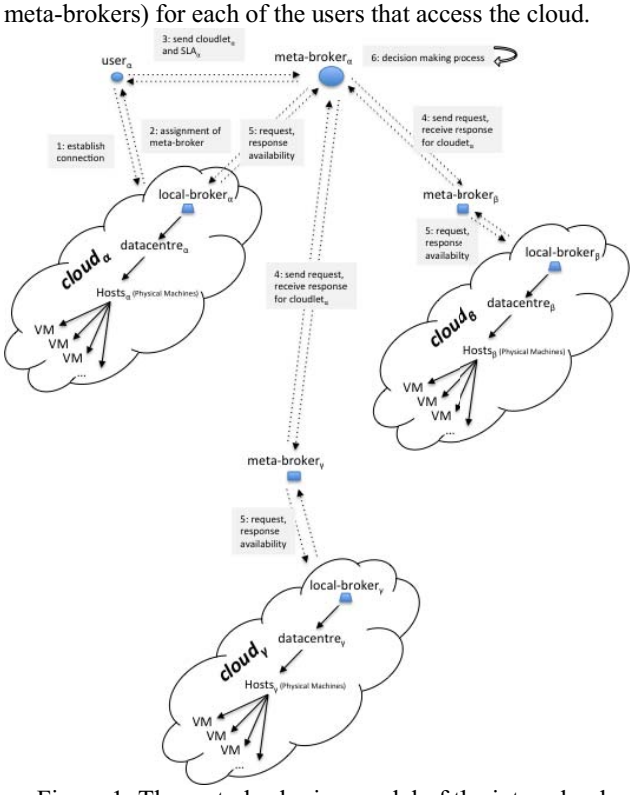

Figure 1: The meta-brokering model of the inter-cloud communication

Figure 1 demonstrates the collaboration scenario of an inter-cloud that is constituted by three clouds named as *cloud*, *cloud*, *cloud* that are sub-clouds of the inter-cloud setting. Specifically, a *user* requests for resources by establishing connection with a *cloud*. Then, cloud assigns to the user a local-broker and a meta-broker following their autonomic generation caused by the original service request from the user.. The local-broker accesses the internal infrastructures (e.g. datacenter hosts and VMs along with provisioning and allocation policies) for resource utilization. The aim behind this decision is to differentiate the internal procedures that are about to be handled by the *local-broker* and the external procedures for requesting resources from the inter-cloud and controlled by the *meta-broker*. Each meta-broker encompasses a meta-registry profile that is an address book of public meta-brokers available to receive communication. In advance the local-broker access acloud datacenter characteristics to utilize internal components e.g. hosts and VMs.

For the period of service submissions the meta-brokers can have information of other meta-brokers (related with the specific scenario) at the service run-time. In addition a meta-broker does not required to transfer its internal knowledge but only to forward a request to inter-connected neighbouring. This perspective offers a high transparency level for the entire cloud since the users are only mapped to their assigned meta-broker, while the last one spontaneously directs the process. The decision for inter-connection is related with the information stored within the meta-registry profile regarding known addresses of meta-brokers. Specifically, we assume that each meta-broker contains a number of meta-brokers assigned by a probability distribution function given by formula (1) where X the number of known meta-brokers, and a and b represent the maximum and minimum number of known meta-brokers.

$$Pr[a \le X \le b] = \int_a^b f(x)dx \quad (1)$$

During the service submission, the meta-broker communicates with the local broker for information exchanging. This includes a total view of the local knowledge. In addition the meta-broker collaborates with other meta-brokers as discussed above. This fundamentally includes the swapping of keys recognizing each other for enhancing the security and trust level. Then each one replies with an acknowledgement message for denoting the acceptance on collaboration. Then the requester meta-broker asks responder for information based on various criteria (e.g. resource availability, heterogeneity etc.). The responder meta-broker sends the information back by executing a matchmaking procedure.

Relevant information that could be exchanged among local and meta-broker is a list of cloud information e.g. datacentre hosts, VMs, jobs (cloudlets) along with the characteristics of the datacentre (e.g. hosts CPU, Memory, number of processing elements (PEs), architecture etc.). Nevertheless, depending on the actual scenario information exchanging could be minimized to the level of resource availability or maximized to the complete internal knowledge analogous to the desired scenario. As the whole

environment is based on the contacted meta-brokers' real-time responses this minimizes the overall information exposition. It should be mentioned that service submissions, can be monitored from an external component and data could be utilized in future for enriching service submissions decision-making.

In this work we aim to an inter-cloud resource discovery management. This includes the clustering of meta-brokers according to service requests criteria. This is closely related to the meta-registry profile that is responsible controlling the clustering operation. Thus, the next section illustrates the meta-registry profile of a meta-broker formed by recording past cloudlet submissions.

IV. THE META-REGISTRY PROFILE

The meta-registry contains a private part for controlling local-broker information and a public part of addresses of known meta-brokers for inter-exchanging information. The private part includes the datacentre information as retrieved from the local-broker as follows:

- The architecture of the cloud datacentre (e.g. Intel)
- The operating system used by the datacentre (e.g. Linux)
- A list with the available hosts
- The cost of the datacentre usage for a) cost per second, b) cost per computational unit and c) per storage or per used bandwidth
- The mips rating of the total mips of the number of the datacentre machines.
- The total pes of the number of the datacentre machines.
- The quantity of the currently idle (non-busy) pes for all the machines of the datacentre.
- The quantity of the currently busy pes for all the machines of the datacentre.
- A case for a broker that requires knowing the status of the submission.
- A case for submitting a cloudlet sent by the broker.
- A case for altering the status of the cloudlet as posed by the broker. This includes the following statuses, a) pausing, b) resuming, c) cancelling and d) updating.
- A case for a broker that requires creating (or overriding in the case that the VM exists) a VM in the datacentre. In such situation the datacentre instantiates a host allocation policy for provisioning VMs to Hosts.
- A case for releasing a VM created previously within a datacentre host.
- A case for handling the VM allocation policy for provisioning VMs to hosts.
- A case for setting the time that the service submission enters the datacentre.
- A case to store the information collected by a datacentre sensor in a list for future usage (e.g. energy consumption).

The public part of the profile contains the requirements specification as introduced by the user in the form of a cloudlet. The requested cloudlet will be bounded within a unique VM designed or instantiated specifically for the particular cloud client demand. The decision behind the VM development is related with the cloud provider as presented in [8], and it is out of the scope of this study. Specifically the public profile includes the following information:

- A unique constant name for each of the users that is identical to the meta-broker id as the ICMS assigns one unique meta-broker per user.
- The input service (job) size measured in the required million of instructions per second (million instructions per second).
- The output file size following the service execution measured in bytes.
- The number of the physical processing elements required for the service submission (e.g. four for a quad-core CPU).
- The identification number of the specific service submission to be identified by the broker.
- The status that denotes the standing of the service submission (e.g. running, waiting or idle).
- The time that the service submission starts execution (measured in ms.).
- The time that the service submission finishes execution (total execution time in ms.).
- The reservation status of the current service submission (yes or no).
- A list that contains the addresses of the collaborated well-known meta-brokers based on the probability formulae (1).
- A case for storing timeline events of meta-brokers life-cycle.

To conclude the profile is formed according to the available internal resources and the requested information. We separate the design into private and public parts for minimizing the information exchanging quantity and keeping internal cloud knowledge isolated from external sub-clouds. The next section illustrates the resource discovery architecture of the inter-cloud meta-brokering model.

V. THE RESOURCE DISCOVERY ARCHITECTURE

The resource discovery architecture encompasses the cloud internal components for service communication along with the inter-collaborated feature. Specifically, the following demonstrates typical sub-cloud internal procedures during a service request from a user.

A. The internal Procedure

Lets assume that there is one inter-cloud setting composed by a number of interoperable sub-clouds each of which is named as c_a where $c_a \in \{c_1, c_2, \ldots, c_a\}$. Individual clouds comprise a number of physical datacenters dc_b - where $dc_b \in \{dc_1, dc_2, \ldots, dc_b\}$ - that constitute the cluster of core resources. Further, each datacenter contains a number of physical machines named as h_c hosts - where $h_c \in \{h_1, h_2, \ldots, h_c\}$ - in such way that CPU cores, ram and storage space of machines integrate the whole cloud computational capacity. In addition, each datacenter dc_b generates a number of local-brokers $lbr_e \in \{lbr_1, lbr_2, \ldots, lbr_e\}$ and

assigns one local-broker per user submission for managing the internal service allocation and execution of a cloud c_a. At last, each of the hosts generate or instantiate a number of VMs for sandboxing the service requirements. Each $vm_d \in \{vm_1, vm_2, \ldots, vm_d\}$ represents the virtualized part of a datacenter host (machine) that eventually contains and executes user service submissions.

B. The interoperable procedure

The interoperable procedure contains the functionalities for achieving inter-cooperation among sub-clouds. Within such system, each datacenter local-broker lbr_e belonging to a datacenter dc_b generates a number of meta-brokers $mbr_e \in \{mbr_1, mbr_2, \ldots, mbr_e\}$ and assigns one meta-broker per user submission and for managing and monitoring the information exchanging among the cloud actors (users, local and meta-broker). Finally, a user (client of the cloud) requests for a service allocation that contains a requirements' specification service (according to the meta-registry) and passes this request to the meta-broker. The last one distributes the request to interconnected meta-brokers that exists within a public profile named as meta-registry. The service (cloudlet component) is then sent from meta- to local-broker and then to the low-level components of the datacenter. This is to say that each cloudlet $cl_n \in \{cl_1, cl_2, \ldots, cl_n\}$ is assigned to a virtual machine vm_d that has been generated to a remote host h_c belonging to a cloud c_a datacenter dc_b.

As presented in figure 1, a request is sent to meta-broker$_b$ and meta-broker$_c$ respectively, then to the internal local-brokers (local-broker$_b$ and local-broker$_c$), where each of which proceeds with matchmaking the cloudlet requirements and the public profile known as the service level agreement (SLA$_a$) for resource provisioning specification. Responses are then sent back to the meta-broker$_a$, including replies from meta-broker$_b$, meta-broker$_c$ and local-broker$_a$ if there are available resources that match cloudlet and SLAs. After that, the decision-making process happened within the meta-broker$_a$ chooses the resource to be utilized and a call for resource allocation is sending again to the selected meta-broker. Resources are then allocated in the form of a VM that generated within a remote host of the selected datacenter. Finally, the user instantiates the VM and executes the cloudlet in a remote sub-cloud.

To conclude in this section we have presented the inter-cloud generic resource discovery happened by the meta-broker. In this phase we assume that various clouds establish connection and exchange information through a newly generated meta-broke. Figure 2 illustrates that procedure by incorporated the following steps.

a) The user initializes connection with a sub-cloud part of the inter-cloud setting through a cloud interface.

b) The cloud sets a local broker that accesses the datacentre knowledge and then forms the meta-registry with information presented in section IV.

c) The local-broker generates a meta-broker which is assigned with addresses of other cloud meta-brokers utilizing the meta-registry.

d) The user gets a message to submit its cloudlet to the meta-broker.

e) The meta-broker accesses the list of meta-brokers and forwards requests to inter-connected clouds for achieving resource discovery.

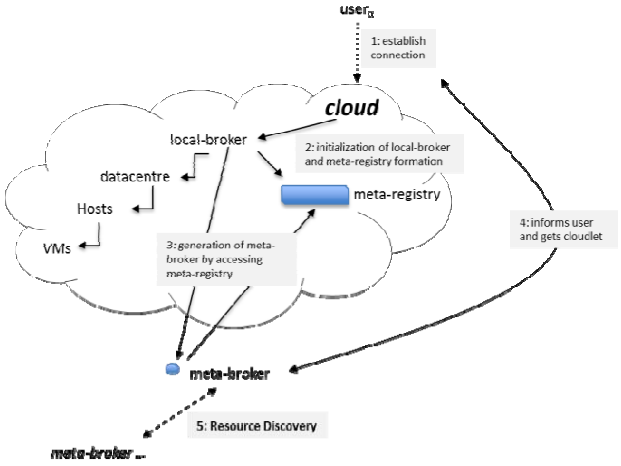

Figure 2: The cloud resource discovery initialization phase.

The next section presents the resource discovery case that includes the clustering of the resources according to previous service submissions.

VI. THE DISCOVERY CASE OF META-BROKERS CLUSTERING

This section presents the resource discovery case of meta-brokers clustering. Specifically, we assume that two clouds (cloud$_a$ and cloud$_b$) establish a partnership connection and are prepared for services exchange according to specific demands. We assume that initialization occurs by the cloud administrators. Then the following steps are taking place for discovering available resources.

(1) A user$_{a1}$ requests resources from a cloud$_a$ (submits a cloudlet$_{a1}$).

(2) The cloud$_a$ assigns a meta-broker$_{a1}$ and a meta-registry$_{a1}$ that contains the meta-registry data (section IV).

(3) At this time instance, there are no existing submissions hence, the cloud$_a$ allocates resources and executes cloudlet$_{a1}$ in a VM$_{a1}$.

(4) Next, a user user$_{a2}$ request resources from cloud$_a$ (cloudlet$_{a2}$)

(5) The cloud$_a$ assigns a meta-broker$_{a2}$ and a meta-registry$_{a2}$ that contains the meta-registry data.

(6) At this time instance, there is one meta-broker (meta-broker$_{a1}$) belonging to the same cloud thus the setting does not send a further request for matchamking.

(7) The meta-broker$_{a2}$ requests clustering from meta-broker$_{a1}$; if cloudlets are identical (according to both meta-registries) a group is made named as meta-

community$_a$. We assume that this group is now formed.

(8) The cloud$_a$ allocates resources and executes cloudlet$_{a2}$ in a VM$_{a2}$.

(9) A user$_{b1}$ request resources from cloud$_b$ and a cloudlet$_{b1}$ is formed.

(10) Cloud$_b$ assigns a meta-broker$_{b1}$ and a meta-registry$_{b1}$ that contains the address of an interconnected meta-broker$_{a1}$ which is chosen randomly.

(11) At this time instance, the meta-broker$_{b1}$ compares cloudlet$_{b1}$ with cloudlet$_{a1}$ (of meta-broker$_{a1}$) and if matchmaking happens meta-brokers are matches and clustered in community$_a$. We assume that meta-broker$_{b1}$ now joins community$_a$.

(12) A request for resource matchmaking is then sent to cloud$_a$ (through meta-broker$_{a1}$) and cloud$_b$.

(13) Following this, the first available resource is allocated and a VM$_{b1}$ is created for cloudlet$_{b1}$ according to resource allocation policy.

(14) Next, a user$_{b2}$ request resources from cloud$_b$ and a cloudlet$_{b2}$ is formed.

(15) Cloud$_b$ assigns a meta-broker$_{b2}$ and a meta-registry$_{b2}$ that contains the address of an interconnected meta-broker$_{a2}$ which is chosen randomly.

(16) At this time instance, the meta-broker$_{b2}$ compares cloudlet$_{b2}$ with cloudlet$_{a2}$ (of meta-broker$_{a2}$) and if matchmaking happens meta-brokers are matches and clustered in community$_a$. We assume that meta-broker$_{b2}$ does not joins community$_a$ and creates a new one community$_b$ that contains only it-self.

(17) Then a request for resource availability is send to cloud$_a$ (through meta-broker$_{a2}$) and cloud$_b$.

(18) Following this, the first available resource is allocated and a VM$_{b2}$ is created for cloudlet$_{b2}$.

(19) At this time instance, there are two communities of meta-brokers namely as community$_a$ (meta-broker$_{a1}$, meta-broker$_{a2}$, meta-broker$_{b1}$) and community$_b$ (meta-broker$_{b2}$)

(20) After a specific amount of time four users request for resource by submitting two cloudlets (cloudlet$_{a3}$, cloudlet$_{a4}$) in cloud$_a$ and two cloudlets (cloudlet$_{b3}$, cloudlet$_{b4}$) in cloud$_b$.

(21) Each cloud assigns a meta-broker and a meta-registry to each user respectively. (meta-broker$_{a3}$ and meta-registry$_{a3}$, meta-broker$_{a4}$ and meta-registry$_{a4}$, meta-broker$_{b3}$ and meta-registry$_{b3}$, meta-broker$_{b4}$ and meta-registry$_{b4}$)

(22) We assume that meta-broker$_{a3}$ is interconnected with meta-broker$_{b3}$, and meta-broker$_{a4}$ is interconnected with meta-broker$_{b4}$.

(23) At this time instance, the request are sent to each other for clustering according to cloudlets comparison, among interconnected meta-brokers as well as between brokers of the same cloud.

(24) We assume that after cross exchanging of information the two communities are expanded e.g. community$_a$ contains a cluster of meta-broker$_{a1}$, meta-broker$_{a2}$, meta-broker$_{b1}$ and meta-broker$_{a3}$ and community$_b$ contains a cluster of meta-broker$_{b2}$, meta-broker$_{b3}$, meta-broker$_{a4}$, meta-broker$_{b4}$.

(25) At this time instance, the meta-brokers does not send the request for resource matchmaking to interconnected meta-brokers as each one assigned to first available resource of a cloud host that other requests from the same community have executed within.

(26) We assume that community$_a$ executes cloudlets in hosts that contain a specification of physical resources (e.g. Intel Xeon with Linux OS, input and output files) and a variety of similar characteristics (e.g. pes, mips bandwidth and cost ranges). Similar, community$_b$ executes a specific cloudlet (e.g. Intel Xeon with Microsoft OS) and other ranges of characteristics.

(27) Finally, the first available resources are allocated according to the clustered meta-brokers resource availability and VMs are created for each of the cloudlet, according to the local resource allocation policy of each cloud.

The above communities are fully decentralized and transparent as the connecting links are transient in the scope of enhancing the resource discovery process.

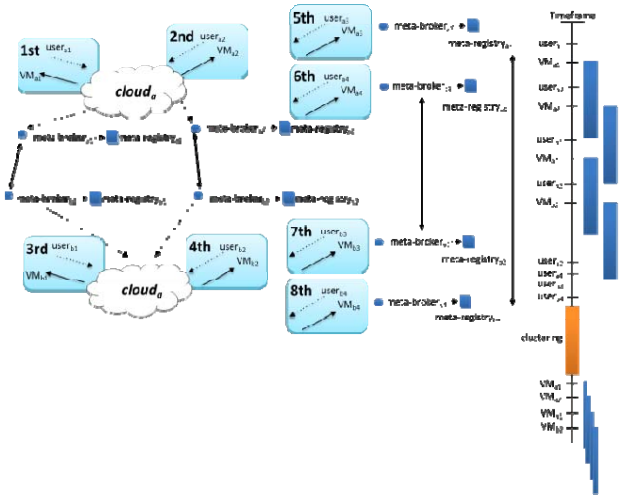

Figure 3: The inter-cloud resource discovery process

It should be noted that after a specific time interval, a procedure checks for a finished meta-broker (VM completion) and drops old meta-brokers that does not have inter-relationships. Figure 2 demonstrates the aforementioned resource discovery steps in brief.

By using the clustering resource discovery method we can achieve two-fold advantages; a) wide resource distribution and at the same time enhancement of the overall resource discovery times and b) transient clustering of

resources according to the job requirements submitted by the users. This occurs because the local-broker does not require searching and matchmaking resources with cloudlets as this is happening by the meta-brokers during the cluster classification process.

The algorithm presents the resource discovery method of clustering according to cloudlet specification including the requirements and operations during this process.

Algorithm: The resource discovery process

Require: $cloud_i$: the cloud i
$datacenter_i$: the cloud datacenter
$host_i$: the datacenter host
$meta\text{-}broker_i$: the user meta-broker
$meta\text{-}registry_i$: the user meta-registry
$cloudlet_i$: the service submission of user i
VM_i: the VM for user i
list: the list with the inter-connected meta-brokers
community: the group of transient connected meta-brokers
Operation: clusteringRequest: the clustering request operation
createCluster: the clustering creation procedure
matchmaking: the cloudlet matchmaking
get: a method to get information
send: a method to send information

1: **for** ∀ $meta\text{-}broker_i$ of $user_i$ **do**
2: meta-registry.get(list) for $cloudlet_i$
3: **for** each meta-broker ∈ list **do**
4: perform matchmaking according to $cloudlet_i$
5: send.clusteringRequest
6: **if** meta-broker.list = **then**
7: createCluster
8: **end if**
9: **end for**
10: **end for**
11: **for** ∀ $meta\text{-}broker_i$ accept submission **do**
12: **if** $meta\text{-}broker_i$ ∈ community **then**
13: request community info
14: **end if**
15: **else** send resource availability request from cloud
16: **end for**
17: **for** ∀ $meta\text{-}broker_i$ accept submission **do**
18: create VM_i in first available $cloud_i.datacenter_i.host_i$
19: **end for**

The next section illustrates a simulation scenario of two communities that aim to compare the traditional service submission (non-grouping) over the clustering discovery method prior to the service execution phase.

VII. Simulation Scenario of Two Communities

This section contains a simulation scenario of multiple users' submissions within an inter-cloud setting of two sub-clouds. Specifically, we explore the average execution time of a multiple cloudlet submission with data extracted from the CloudSim simulation framework [11]. The last one offers a simulation setting for designing and experimental analysis of cloud infrastructure. In our setting we extend the simulation configuration to an inter-cloud system of multiple meta-brokers according to architecture of section V. Within such setting we run a simulation of multiple users that submit multiple cloudlets in different timeframes of the simulation life-cycle. Figure 4 demonstrates the enhancement of the total execution times as well as the trend lines of the experiment for groups that contain from 4 to 40 users. Similarly, figure 5 presents the comparison of cloudlets average execution times for users 4 to 24. It is apparent that the clustering approach over-performs the traditional non-clustering resource discovery method.

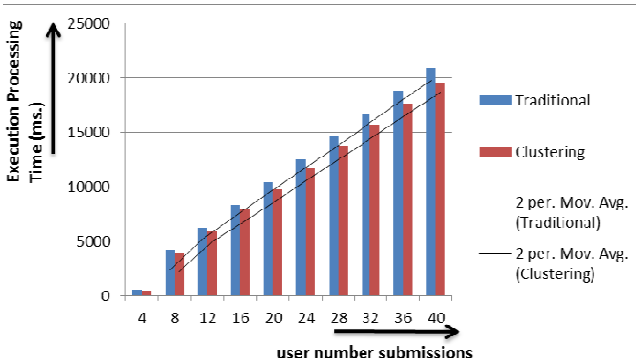

Figure 4: The comparison of cloudlets processing times for 4 to 40 users' submissions.

The actual comparison is based on times extracted from a traditional specification (non-clustering) over the clustering resource discovery.

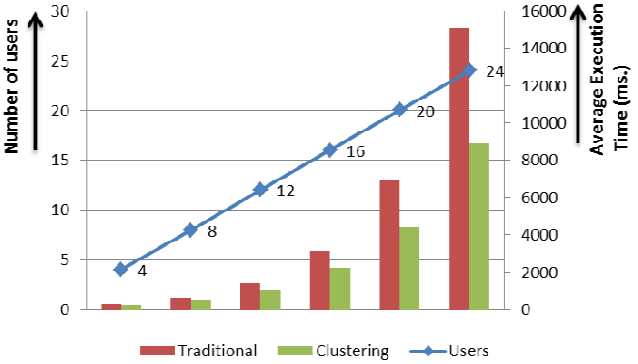

Figure 5: The comparison of cloudlets average execution times for 4 to 24 users' submissions

In this case we develop two communities that execute identical jobs. The service specification has been configured using the pattern of Table 1.

Table 1: The basic simulation pattern

Id	Specs	Cloudlets	VMs	Average Exec Time	Last Proc. Time
1	com_a	100	10	88	160
2	com_b	100	20	48	80
3	com_a	100	10	88	160
4	com_b	100	20	48	80

Specifically, table 1 presents information regarding the id (the user and the sequence of submission), the specs (the community that a cloudlet belongs-according to hardware requirements), the cloudlets number, and the VMs that represent the number of required VMs (e.g. user 1 will execute 100 services in 10 VMs – so the user will pay only for the life-cycle of 10 VMs). At last the average execution

time and the last processing times will be the benchmarks for comparison.

Figure 4 presents that by utilizing the clustering method the processing times are reduced by an amount (10 ms. per user submission) due to the internal matchmaking procedure due clustering. It is apparent that the clustering case offers an efficient service execution time in terms of overall processing time. Also for multiple user submissions we use the same specification of table 1. For example for eight users we use ids 1 to 4 and after the fourth user completes we execute 1 to 4 again. Using the clustering method we have achieved an improved scalability as more loads are handled more efficiently. In addition, the resource availability is higher as the actual processing times end service execution earlier. Finally the transient formation of virtual groups offers fully decentralization, transparency and dynamic-ness mainly due to their relation to the service submission specification. That is to say the characteristics of a job are important to be analyzed during resource discovery for offering an efficient resource orchestration setting.

VIII. CONCLUSION

This work presented the inter-cloud resource discovery method that offers an efficient clustering of meta-brokers for the enhancing of decision making in interoperable cloud partnerships. By presenting the inter-cloud model and the resource discovery architecture we have concluded to a model for grouping meta-brokers according to user submission specification. The study contains the steps for achieving resource discovery by matchmaking cloudlets instead of internal characteristics e.g. distances or weights of interconnected paths. Our simulation experiments show that this solution overcomes traditional settings as it has achieved improved execution times in high peak workloads.

The great advantages of meta-brokers clustering are the high tolerance in failures (if one meta-broker fails the rest of the group keep the same characteristics) and the run-time decision making process. In addition, it offers improved performance, due to the better processing times as observed in Section VII. At last, high availability, scalability is obtained by utilizing a fully decentralized and dynamic model. However, this solution is at its primary steps and requires to be extended in order to achieve other crucial components e.g. resource availability and appropriate policies. Relevant issues like, group failures, or complexity to the huge number of various demands could raise important issues. For that reason we have implemented a selection strategy of groups according to ranges and not actual values.

In addition, an issue that required being resolved is the transparency of the setting in terms of cloudlet exchanging among meta-brokers. This is to say that some of the meta-brokers are not allowed exchanging internal information during run-time. In the case of large scale communities, we can adapt peer to peer search algorithms as we presented in [9] for further enrichments. To conclude, this is the initial study to illustrate the basic resource clustering method to enhance resource discovery based upon user requirements.

ACKNOWLEDGEMENT

This paper is an outcome of the collaborated research effort among the University of Derby, UK and the University of Applied Sciences of Western Switzerland (Fribourg), Switzerland. The first author acknowledges the Erasmus agreement and would like to thank both institutes for their support.

REFERENCES

[1] Bessis, N., Sotiriadis, S., Cristea, V., Pop, F., 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.

[2] Bessis, N., Sotiriadis, S., Cristea, V., and Pop, F., 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 .

[3] Buyya, R., Ranjan, R., and Calheiros, R. N., (2010) InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services, Algorithms and Architectures for Parallel Processing (2010), Volume: 6081/2010, Issue: LNCS 6081, Publisher: Springer, Pages: 13-31.

[4] Ma, Y., Gong, B., and Zou, L. 2008. Resource Discovery Algorithm Based on Small-World Cluster in Hierarchical Grid Computing Environment. In *Proceedings of the 2008 Seventh International Conference on Grid and Cooperative Computing* (GCC '08). IEEE Computer Society, Washington, DC, USA, 110-116.

[5] Iamnitchi, A. and Foster, I., T.. 2001. On Fully Decentralized Resource Discovery in Grid Environments. In *Proceedings of the Second International Workshop on Grid Computing* (GRID '01), Craig A. Lee (Ed.). Springer-Verlag, London, UK, 51-62.

[6] Wang, M., and Xu, H. 2006. Small-World-Clustering-Based Grid Resource Serach Algorithm, Journal of Beijing University of Posts and Telecommunications, Vol. 29, No. 1, Feb 2006.

[7] Ali, K., Datta, and S., Aboelaze, M. 2005. Grid resrouce discovery using small world overaly graphs[J]. Electrical and computing Engineering, 2005. Canadian Conference, 1010-1013

[8] 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.

[9] 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.

[10] Sotiriadis, S., Bessis, N., and Antonopoulos, N., 2011. Using Self-led Critical Friend Topology Based on P2P Chord Algorithm for Node Localization within Cloud Communities. In *Proceedings of the 2011 International Conference on Complex, Intelligent, and Software Intensive Systems* (CISIS '11). IEEE Computer Society, Washington, DC, USA, 490-495.

[11] 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.