Modelling Requirements for Enabling Meta-Scheduling in Inter-Clouds and Inter-Enterprises

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

Abstract— **Cloud computing provides a promising paradigm for the deployment and utilization of online resources including hardware and software services by Internet users. In such an e-infrastructure environment, the scheduling of user-defined tasks is always considered as a complicated part of the overall system modelling. Specifically, in the case of inter-clouds and inter-enterprises scheduling optimization is fundamentally important for achieving the best possible capacity in terms of resource utilization. Thus, existing approaches that consider system dynamics, interoperability and heterogeneity issues become important aspects for providing advanced scheduling decisions. In this work, we survey some highly dynamic meta-schedulers that are suitable for enterprises using grids and/or clouds. Our intention is to elicit the characteristics and produce a model encompassing the architectural requirements that will enable high meta-scheduling performance in inter-cooperative e-infrastructures.**

Keywords: Cloud; Inter-clouds, Inter-enterprises, Federated Clouds, Meta-schedulers, Community Aware Scheduling

I. INTRODUCTION

During the last decade, the cloud computing paradigm became one of the most promising technologies for achieving distribution of resources in a wide area scale. However, as today's trends tend users to challenge the cloud elastic services, a significant approach namely inter-clouds [7] that enables inter-cooperation between clouds, has been introduced as to improve the quality of service offered by the capacity oriented clouds. Specifically, authors in [1] discuss the inter-cloud vision and suggest that the major aim is to facilitate the auto-scaling of resources among various cloud infrastructures for exchanging services.

In parallel to our previous work [15] we have highlighted the need for an aggregated view of sharing cloud resources in order to enable inter-cloud provision. In particular, the work [1] suggests that the herein highlighted shortcoming (the lack of a no service coordinating resource sharing among separate clouds) reduces considerably the quality of service while at the same time increases the administrative cost (maintenance and support). A notable example is the case of Amazon, a cloud leader vendor [15], which cloud data-centres don't support collaboration with each other and leave users with the option to select their desired cloud sub-system based on their geographical location; this may decrease the quality of service and scalability.

Recently, some cloud providers aim at enabling an inter-cloud provision, by utilizing a pool of distributed cloud resources in order to improve their quality standards. Specifically, joint clouds form a pool of collaborated sub-clouds or sub-resources (e.g. grids) similar to a dynamic distributed system. Within such an e-infrastructure, one of the most fundamental aspects, which emerges architectural design issues is the scheduling algorithm [14] that deals with the selection of resources considering the characteristics of the actual system (centralized or decentralized) as well as the requirements of the desired scenario.

In general, various taxonomies of schedulers have been introduced, including schedulers for operating system (OS), high performance computing, parallel and meta-computing [17]. Among all these, the task of scheduling in the meta-computing has been proven to be the most complex [2][10] mostly due to the involvement of a mixture of local resource management systems (LRMS). In addition, unforeseen situations, i.e. the highly dynamics (fluidness) of the meta-computing environment in which resources can join and leave unexpectedly, can lead to increased complexities [14].

In [15] we have surveyed a number of highly dynamic schedulers applied within meta-computing environments, i.e. grids. In this paper, we aim to identify the requirements for an inter-clouds meta-scheduler, which will enable higher performance in terms of the quality of service to be offered by the involved clouds. Within this context, Section II presents the need for the inter-cloud meta-scheduling, and Section III introduces the problem statement. The rest of the paper is organised as follows. Section IV discusses the functionality of the dynamic algorithms and Section V the model of requirements drawn from the literature study. Finally, Section VI presents the conclusion and the future work towards to inter-cloud meta-schedulers.

II. THE NEED FOR INTER-CLOUD META-SCHEDULING

The cloud paradigm share several commonalities with other technologies including grid and utility computing by combining their most important characteristics in order to offer a variety of services i.e. infrastructure as a service etc. Clearly, the key features of each one of them contribute towards additional complexity when considering a (meta-) scheduling approach [17]. For example, the grid characteristic of resource geographical distribution and the use of virtualization technology in utility computing highlight new requirements for clouds. Similarly, in case of inter-enterprises of cloud-to-cloud and/or cloud-to-grid, also known as inter-clouds, new requirements are required for achieving optimum scheduling performance.

In particular, the need for an inter-cloud meta-scheduling approach has been introduced in [15], which provides a detailed taxonomy of a number of meta-approaches for identifying requirements, characteristics and classify solutions in terms of most and least suitability for inter-clouds. Specifically, we have highlighted the growing interest in inter-cloud computing which includes endeavour of the biggest vendors in the area such as HP, Intel, Yahoo, etc. [1]. However, their innovative efforts have led to the establishment of a federation of collaborated clouds with joint initiatives. The inter-cloud vendor-oriented approach has a specific control plane rather than of a setting based on future, sustainable standards and open interfaces.

Having said that, inter-clouds can be considered as a wide research effort in which issues such as resource discovery, allocation and scheduling are quite the most important. In this work we only focus on the job scheduling aspect for distributed environments (grids and clouds) so deliberately, we aim to identify schedulers that are easy to adopt within inter-clouds e-infrastructures.

However, one of the most important criterions for our selection is the dynamics of a system as the unpredictability of resources is high [8][11][13] thus, the dynamic-ness of a meta-scheduling approach forms the basis of our research. Specifically, static-ness in scheduling is defined as the approach in which all the decisions done prior to the execution of the schedule. On the other hand, dynamic-ness allows decisions during the execution time and thus, it enables the consideration of unforeseen situations that may occur in large-scale, fluid e-infrastructures. Several static schedulers have been discussed in [2][15] yet in the most of the approaches these have used as the basis for the development of more complex dynamic schedulers, which are discussed in [8][10][17]. In the following section, we describe the problem statement of the present study, which is based on the dynamically changed nature of an inter-cloud e-infrastructure environment.

III. Problem Statement

During the past years scheduling within uncertain environments in terms of scale (e.g. grids) has been extensible studied. A great variety of scheduling algorithms (centralized or distributed) have been proposed by [3][4][5], aiming to a more flexible and efficient operation. However [16] suggest that when things come to scenarios and use-cases aiming at testing a realistic solution it turns to be impossible to design such tools, mainly because of important characteristics such as support for dynamic scheduling are not considered. *In addition, the clear problem is that the actual requirements are not known in advance, which may cause a change of the initial conditions and chosen parameters*. Thus, the strategy for developing such solutions should be fully automated, and flexible in terms of considering *dynamic metrics* as much as possible. In this work, we aim to identify those metrics on the basis of a literature study about two meta-scheduling approaches. In particular, those metrics are closely related to the distributed meta-scheduling theme as is concerned with the distribution of jobs across independent sites in distinct administrative areas. The decentralization aims to overcome the common problems such as the single point of failure and the bottleneck for the cloud environment as the central instance has a solely responsibility for handling all jobs and request. Thus, we select the distributed meta-scheduling approach where various LRMS deploy their own meta-scheduling strategies for job delegations and executions. We continue the present study by discussing the most common meta-approaches for identifying possible inter-clouds algorithms.

IV. The Algorithms

In [15], we have presented a state-of-the-art review of 18 meta-scheduling technologies. The purpose was the analysis of possible solutions for determining meta-scheduling to inter-enterprises including inter-clouds. Furthermore, we have discussed the inter-cloud needs and requirements and we have correlated the characteristics of meta-schedulers to inter-cloud requirements. Throughout the review of the functionality of approaches, we have recognized the need, which eventually leads to the modelling of a forthcoming and novel meta-scheduling algorithm suitable for inter-clouds. Thus, we have concluded that in particular, 6 works out of 18 have been identified as the most suitable candidates to used in an inter-cloud scenario (readers are pointed to [15]), while 2 out of 6 presented in [13][8] offer the most related significant advantages. This suggestion has been concluded after a correlation of current meta-approaches characteristics to the future inter-cloud requirements.

Different from all others, both solutions [13][8], offer decentralized meta-schedulers that evaluate static and dynamic scenarios of total and partial knowledge of the resource pool and use previous recorded history data of work delegations to improve their scheduling decision. In addition, experimental results have shown significant improvements of the selected benchmarks when the dynamic scenarios are applied and compared with the static and non-dynamic ones.

In particular, [13] presents four scenarios of decentralized settings in which meta-schedulers have partial and full domain knowledge in dynamic job submissions without information exposing. Also, this approach offers advance reservation mechanism and transparent user mapping while compares results and selected benchmarks which shows a remarkable meta-scheduler performance when an unpredictable situation occurs.

Figure 1 demonstrates the relational model of the most important features that are treated as metrics for defining the dynamically changed attributes. In brief the dynamic objective, which represents the actual jobs submissions, happens in a fully decentralized environment which uses history records without information exposing. Testing includes scenarios of fully and partial knowledge of the resource pool, as well as deployment of advance reservation mechanisms. In parallel to [13], [8] presents an advanced solution that evaluates four scenarios of centralized and decentralized approaches in full and partial resource pool knowledge scenarios. Specifically, [8] suggests a model for improving [13] functionality by incorporating real-time processing of information. The aim herein has been changed, as it is to achieve a globalized performance and not with the view of improving each individual participant benchmarks.

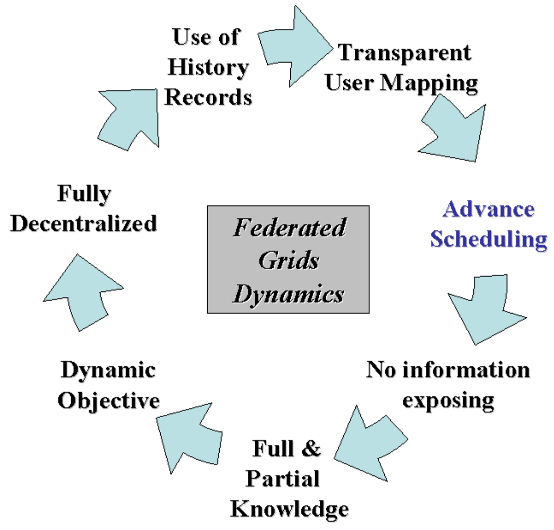

FIGURE 1: DYNAMICS OF FEDERATED CLOUDS

In addition, instead of advance reservations a re-scheduling concept is introduced. The authors conclude that an overall improvement of the system performance happens when considering the dynamics as the meta-scheduler. During re-scheduling, real time information exchanging occurs without internal data exposition. Figure 2 illustrates the dynamics of the [8] meta-scheduling algorithm by adding extra features to the [13] solution.

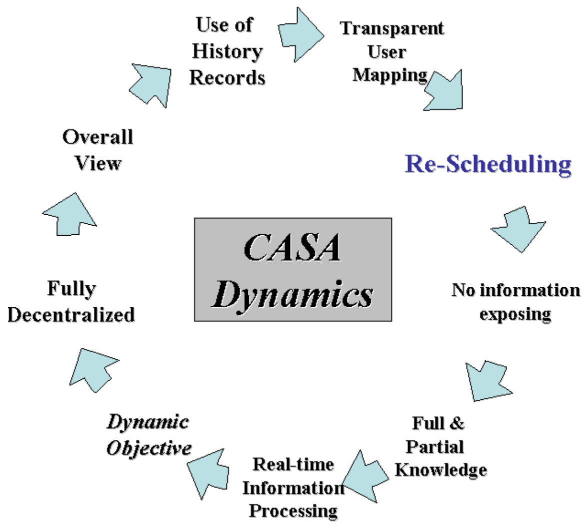

FIGURE 2: DYNAMICS OF THE CASA

The following section presents a detailed discussion of the functionality of a) the Federated Grid Algorithm [13] and b) the Community-Aware Scheduling Algorithm (CASA) [8] with the view to identify the design issues and modelling requirements for inter-clouds meta-schedulers.

A. The Federated Grids Algorithm

This approach provides an alternative to the centralized solutions by proposing the deployment of a meta-scheduler on the top of each grid infrastructure (LRMS) within a federated grid. Next, we discuss the algorithm principle, the objective algorithms (scheduling algorithms), the design and implementation issues, as well as the results from the compared benchmarks.

1) The algorithm principle

[13] presents a meta-scheduling strategy for federated grids as the evolution of the grid computing. Particularly, the work aims to the expansion of the grid functionality as to allow resource sharing among several grid infrastructures. The model is a generic decentralized approach which places meta-schedulers on the top of each grid system. In their study, authors propose four algorithms to compare the performance of the different grids forming a federation.

Their decentralized scheduling approach includes that a meta-scheduler will be placed on the top of each grid infrastructure that implements appropriate algorithms. Also the users mapping will be fully transparent. Each of the four algorithms of the meta-scheduler aims to increase the performance while reducing the make-span. Finally, the mapping of jobs to resources doesn't require configuration information. For implementing each of the mapping strategy the approach use the GridWay [13] a meta-scheduler for grids. Specifically, a performance model as described in [13] is applied to the GridWay to obtain results from the objective functions (makespan and performance of resource) of each of the four algorithms.

The GridWay's current scheduling policy implements the Normal algorithm. It is a simple mapping policy that maps a number of `DISPATCH_CHUNKS` of unscheduled jobs to internal or external resources each a time of `SCHEDULING_INTERVAL` in seconds. The algorithm maps the next job to the first internal resource, in the case of no availability the job is mapped to an external. *This basic functionality is relying upon the decision of available free nodes without considering past performance data of participating nodes.* Then, the algorithm uses a migration control to check if the job will start its execution before the SUSPENSION_TIMEOUT value is exceeded, in that case the job is scheduled again. The authors examined the effect of the algorithm and claim that this is a simple grid scheduler that doesn't provide significant results under the selected scenario. The following presents the four mapping algorithms.

2) The Objective Algorithms: SO, DO, SO-AS, DO-AS

Herein we present an elaborated discussion based on the four objective algorithms discussed in [13] including static and dynamic objectives as well a combination of each approach to the advance scheduling strategy.

a) <u>Static Objective (SO)</u>

The variable that denotes the number of jobs to be submitted to each of the grid infrastructure is called an objective, and it aims to maximizing the throughput of internal resources while at the same time the make-span value remain untouched.

The SO algorithm as denoted by its name is static, thus performance measurements are obtained in the form of linear equation for each participant by training the testbed using the Normal algorithm. Once they have those results, they decide the number of jobs that should be submitted to each participant, in an off-line fashion (total domain knowledge), for increasing the throughput of internal resources without increasing the makespan. Then the algorithm is updated and executed. Specifically, the SO maps `DISPATCH_CHUNK` of unscheduled jobs to internal resources every `SCHEDULING_INTERVAL` seconds. If there aren't any available internal resources the SO schedule jobs to external resources.

b) Dynamic Objective (DO)

In this case the DO calculates objective dynamically instead of the off-line procedure of the SO algorithm. In particular, the DO algorithm first determines the performance of each internal resource. Then, it calculates the linear relations of the whole federated grid by viewing it as an aggregation of participant grids. *Specifically, the DO algorithm uses the jobs delegations to internal resources (previous history records) to calculate the number of jobs to be submitted to external resources*. Finally a new event is programmed for updating the next objective.

The advantage of DO, when compared with SO, is that *the first one can be easily be adapted to a large scale and complex grid* while at the same time it behaves similar to SO, by mapping a number of `DISPATCH_CHUNKS` of jobs to resources every `SCHEDULING_INTERVAL` seconds.

c) Static Objective - Advanced Scheduling (SO-AS)

The SO-AS combines the static objective and the advance scheduling strategy by queuing jobs to target resources in advance without waiting for free resources. In this way*, the latencies are avoided while at the same time the performance is increased.* In particular, similar to aforementioned objectives the scheduler maps a number of `DISPATCH_CHUNKS` every `SCHEDULING_INTERVAL` in seconds. So, firstly, the SO-AS algorithm determines what jobs of the `DISPATCH_CHUNKS` are `internaljobs` and creates a number called `toIntenal`. This number is proportional to the number of jobs that the SO algorithm decides that have to be submitted to internal resources.

After that, the next job is mapping by the AS which checks if the `scheduledjobs` keeps less than the DISPATCH_CHUNK of jobs. In the case of an internal job and available internal resources the job is scheduled to one of them. The `scheduleToInternalResource` queues to a limit of number of nodes multiplied with a max running resource factor. In the case of an external job the algorithm avoids the situation in which the federation of grids can receive jobs from another participant and schedules jobs to internal resources. Finally a new event is programmed for the next objective update.

d) Dynamic Objective - Advance Scheduling (DO-AS)

The DO-AS algorithm is similar to the SO-AS with the only difference that calculates the new objective dynamically. Also the advance scheduling strategy denotes that the scheduler doesn't wait for a free node and queues jobs in advance in the target resources.

3) The design, the implementation and the test scenario of the objective algorithms

The authors implement five versions of the GridWaySim that differ in the mapping strategy namely the Normal, So, DO, SO-AS and DO-AS and perform simulations using the GridSim testbed. All the aforementioned versions have the same configuration, the same number of users that submit their jobs to the same time to the same broker.

The test scenario contains two grid resources, the DSA (Distributed Systems Architecture) and the LCG (LHC Computing Grid). DSA represents the grid group of the "Universidad Complutense de Madrid", while LCG testbed represents the "Large Hadron Collider (LHC) Computing Grid". In this scenario, a DSA internal user could submit a job to the DSA GridWay broker, and the job can be either executed by the DSA interval resources or the LCG external resources. Similarly jobs submitted to the Globus WS-GRAM interface of the LCG GridWay are from external users. They demonstrate in this scenario a common situation on which at the same time use its internal resources.

4) 7.2.4 Simulation Entities, Parameters and Results

At the simulation time the GridWaySim generates three users, two GridWay meta-schedulers, one DSA testbed, one LGC testbed and one Workload trace. Specifically, the two meta-schedulers are required because the scenario demonstrates the inter-connection of the DSA and the LGC Grid, so one meta-scheduler for each grid is required. The testbed resources are represented in the test scenario and they are called DSATestbed and LCGTestbad respectively. The actual experiment is a collection of 550 equal jobs, which as suggested by [13] the sample represents a medium-sized grid experiment. The authors discuss their experimental results based on the level of saturation of the LCG resource, when at the same time knowing the number of free available PEs. Thus three users submit their experiments as follows. User_0 submits the experiment to an ideal scenario of low saturation with free available PEs. User_1 submits the experiment to a medium saturation scenario. Finally, User_2 submits the experiment to a high saturation scenario of limited number of free available PEs. Then the authors present the number of jobs executed on the three scenarios, which represent the objectives of the experiments. In the case of different algorithm the objectives are not similar mainly due to the off-line functionality that offers a full view of the performance of the resources as happened in the case of SO and SO-AS. On the other hand, in the case of DO and DO-AS, the knowledge of the same performance is partial. At last, DO and DO-AS differ in the advance scheduling strategy, which doesn't wait for free nodes at the time of job submission. The results extracted from [13] follow:

a) The Normal and DO algorithms behave normally and as saturation is increased more of the jobs are executed in DSA.

b) SO-AS and SO show a different behaviour and they produce poor results because of the "long periods on which single jobs cannot be executed in LCG".

c) DO-AS behave also differently and calculates very similar objectives for three levels of saturations.

Then the authors present the completion time for each job to each level of saturation and with this way they offer a global view of the five algorithms as follows:

a) Normal, SO and DO submit jobs when there are available nodes thus when saturation is increased they need more time for completing the experiment.

b) DO-AS queues jobs in advance without waiting for free available nodes, thus authors conclude that the performance is more consistent as the results show almost identical because the queue time in the LCG is practically null.

c) The most notable results are from User_2 which the queuing of jobs is done in advance without waiting for free PEs.

d) Finally the authors suggest that the combination of the performance model with the scheduling in advance reveal the best mapping performance.

B. The Community-Aware Scheduling Algorithm

The Community-Aware Scheduling Algorithm (CASA) is a two phase scheduling solution which contains a set of five heuristic sub-algorithms. The functionality of the CASA is elaborated on the basis of work shown in [8] starting with the algorithm principle, the sub-algorithms explanation, the simulation environment, and the results.

1) The algorithm principle

The CASA assumes that a number of N nodes exists in a grid environment, wherein $N = \{n_1, n_2, \ldots, n_n\}$, in which some of them have job submissions. Within this setting the nodes that assign jobs to others are called *initiators* and the nodes receiving requests are called *responders*. Each time a job called $j_{,}$ is sent from an initiator α to a responder the following characteristics are considered:

a) The required processing elements (PEs) including CPU, memory, etc. of the job denoted as $j^{pe}_{,}$

b) The estimated execution time denoted as $j^{length}_{,}$

c) The estimation of the processing elements, also known as million of instructions per second denoted as $j^{mips}_{,}$

d) The job load calculation based on previous assumptions calculated by the formulae $load = j^{pe}_{,}\ j^{length}_{,}\ j^{mips}_{,}$

The CASA also considers the unpredictability of the grid system by highlighting that the dynamics such as, the local job submission distribution, the local job characteristics; the job status and resource usages are impractical to be distributed. Thus, a more advanced solution is to retrieve required information at the runtime from real workload traces as done in CASA. In particular, the algorithm is a phase-by-phase solution which consists of two sub-scheduling algorithms namely as the job submission phase and the dynamic scheduling phase. Starting with the job submission phase the algorithm decides the strategy for submitting jobs to candidate nodes. Specifically the following steps contain a step-by-step explanation of the submission phase functionality.

a) A node n_a receives a job $j_{,}$ from the local user n_a is the requester and generates a `request` message using the algorithm $h_{request}$

b) The job characteristics (PEs, heterogeneity) are including in the message an interface $it_{neighbors}(n_a)$ is invoked in order locate a list with discovered remote nodes the `request` message is distributed to each of the remote nodes of the list each responder node receiving the `request` message invokes the h_{accept} algorithm to decide the job execution or not including node's capabilities and administrative characteristics.

c) In the case that delegation is decided, an `accept[R]` message $acc^{R}_{,\,,}$ is generated and sent back to the requester the `accept` message contains the estimated response time of the responder node n requester collects all the `accept[R]` messages and invokes the h_{assign} to select the remote node for delegation.

d) The algorithm h_{assign} adopts a probabilistic approach for selecting nodes without greedily selecting the best ones which will lead to starvation.

e) Finally the node n is selected and an assign message is send by the h_{assign} which encloses all the related data.

Next, we present a detailed discussion of the sub-algorithms.

2) The sub algorithms

The dynamic scheduling phase is a complementary action for adapting to dynamic and unpredictable changes of the system. Specifically, each node n_γ in the grid contains a local queue with jobs submitted either from the local user or remote nodes. First the $h_{resched}$ algorithm is invoked in $t^{interval}$ intervals to find the jobs with the longest waiting times of the local queue. Afterward the interface $it_{neighbors}(n\ )$ will be invoked to find a list with remote nodes and the $h_{resched}$ algorithm generates an inform message $inf_{,}$ which contains the rescheduled job including its characteristics (PEs, estimated execution time, job profile). After that the responder node receives the message and invokes the h'_{accept} algorithm to decide whether or not the job delegation will be accepted. The h'_{accept} works similar to the h_{accept}, except one difference, there should be worthwhile benefit from rescheduling. If the n accepts the job an `accept[I/` message $acc^{I}_{,\,,}$ is send to n. The n collects and compares the `accept[I]` messages for $job_{,}$ and selects the appropriate node for rescheduling the job.

In the following we detail the CASA sub-heuristic algorithms namely as $h_{request}$, h_{accept}, h_{assign}, $h_{resched}$, and h'_{accept}. The $h_{request}$ algorithm is launched every time a local user submits as job called $job_{,}$ to a $node$. At this point the job submission phase is initiated with the aim of locating the proper node from the scope of the overall grid. Specifically, the $it_{neighbors}$ interface is invoked in order to fetch a list with the addresses of remote nodes. Then the $h_{request}$ sends a message with the characteristics of the job, including, $j^{pe}_{,\,,}\ j^{length}_{,\,,}\ j^{mips}_{,}$ to each remote node.

The h_{accept} is invoked in each remote node and represents the delegation acceptance algorithm. Then, jobs descriptions are matching with the remote resources characteristics, e.g. heterogeneity and number of processing elements required. Finally, the responder generate an `accept[R]` message along with an estimation of the execution time is send back to requester node.

The h_{accept} calculates the estimated job response time by summing up the total time of the queue and the estimated time of the job. The local job queue contains a sequence of already received nodes waiting for execution. The shadow job queue contains a sequence of already `accept[R]` messages probability along with the job characteristics profile. The h_{assign} algorithms selects the responder node which offers the shortest response time and delegates the $job_{,}$ to the node. Before that, several accept messages have been received from remote nodes. In the case that the same nodes are greedily selected from the requester then a probabilistic procedure happens.

The $h_{resched}$ algorithm for checking periodically which of the already queues jobs could be reassigned to other nodes for improving the overall performance and grid resource utilization. The basic idea is to select the appropriate jobs, which are already queued to different nodes and re-assign them to various nodes.

The h'_{accept} is the algorithm for requesting rescheduling acceptance from the remote nodes. Specifically, when a node receives a reschedule request estimates the response time for each job waiting in the local queue.

3) The simulation scenarios and the benchmark results

The CASA scheduling sub-algorithms has been evaluated in four different scenarios for running jobs in distributed and heterogeneous resources. Those are the local scheduling, the centralized meta-scheduling, the decentralized meta-scheduling with global knowledge and the decentralized meta-scheduling with partial knowledge scenario as follows.

1. Ind: This is the simplest case in which each node in the pool receives the same amount of jobs which are eventually handled by the LRMS of each node
2. Cen: In the centralized scenario the jobs are submitted to a unique centralized meta-scheduler which has a detailed knowledge of total PEs and PEs at any given time of each remote node. The scenario uses the BestFit heuristic algorithm to demonstrate the optimization scenario regarding to the benchmark metrics. Those are the job response time, job waiting time and job slowdown.
3. Dec-G: In the decentralized scenario each node adopts a complete knowledge of the remote nodes. Also, the nodes don't have a detailed knowledge of PEs of other nodes, so the CASA will be evaluated for job delegation behaviours.
4. Dec-P: In the decentralized scenario with partial knowledge of the resource discovery service each node has knowledge of six remote nodes for delegating jobs.

The observed results are illustrated as follows:

1. <u>*Job success completion*</u>

a. Ind: in the independent scenario jobs require different operating system cannot be delegated, thus they fail. Also the completion rate is 67% on the arrival nodes.

b. Cen: the centralized meta-scheduler has full control of nodes. The completion rate is 99% when 3% of jobs are executed from nodes hosting the centralized meta-scheduler

c. Dec-G: the decentralized meta-scheduler with complete knowledge allows nodes to find other remote nodes during the job submission phase because of this global resource discovery service. It has been observed a 99% of completion rate with 6% of jobs to be executed by the arrival nodes.

d. Dec-P: the partial knowledge of the resource discovery service of the decentralized meta-scheduler (only six nodes) slightly decreases the completion rate to 95%. A 14.7% of jobs are executed by the arrival nodes.

2. <u>*Resource Usage*</u>

a. Ind: in the independent scheduling scenario the 32% of submitted jobs are not executed by any resource. Also, uptime when compared with centralized and decentralized (dec-g, dec-p) is lower wherein 99% are executed.

b. Cen and Dec-G: both scenarios share the same average resource uptime, thus decentralized and dynamic CASA can really promote a resource usage level similar to the BestFit heuristic algorithm. *The great advantage is that centralized BestFit request resource information in contrast with CASA that doesn't request either information or control authorities data, thus offering great flexibility.*

c. Dec-P: The resource utilization is slightly decreased about 6% when compared with the complete knowledge of the resource discovery service scenario.

3. <u>*Job response time, slowdown and job waiting time*</u>

a. Ind: in the indecent scenario the average response time (the time between jobs' arrival and return time) is 10,608 seconds. The average job slowdown (the ration when comparing jobs response time with the processing time) is 135. The job waiting time (the waiting time of a job until execution) is 5459 seconds.

b. Cen: in the centralized scenario the average response time is 11,665 seconds, a little more because in Ind scenario 32% of jobs suspend their execution and then fail due to the non prioritized job requirements (e.g PEs). The job waiting time is better when compared with the Ind scenario (around 1000 seconds).

c. Dec-G: In the global knowledge of the decentralized scenario the response time is 7790 which is less than the both aforementioned Ind and Cen. Also, the job slowdown is 2/3 of the decentralized. Finally the global resource discovery service is significantly improved.

d. Dec-P: in this scenario with the partial knowledge (only 6 nodes) the job response time is 6.598 seconds and the waiting time is 764 seconds which is the most optimized of all scenarios. The results show that this solution is better than the independent and centralized scenarios.

To conclude the CASA is based on nodes' real time responses and behaves dynamically without exposing information from resources. *The simulation result shows that job slowdown, waiting times, response times and messages overhead values have been significantly improved.*

V. Modelling the Dynamic Metrics of Inter-Cloud Meta-Scheduling

Having discussed both CASA and FD, we conclude to the dynamic requirements of meta-scheduling in inter-clouds with the aim of providing an aggregated workload balancing mechanism.

When correlating and comparing the Federated grids meta-scheduling algorithms and CASA, we conclude that there is a number of positive features that are applicable and suitable for the inter-cloud meta-scheduling theme. Firstly, the federated grid scheduler [13] in the dynamic scenario (DO) shows an easily adapted behaviour in the large scale of the grid while the results shows the same amount of job mappings with the static scenario (SO). Thus, latencies are avoided and performance is increased. In addition, the advance scheduling is proven valuable as it uses previous job delegation data for queuing jobs to target resources in advance. In general the best results are coming from the dynamic objective with the advance scheduling (DO-AS) scenario. Herein jobs queuing happen in advanced without waiting for free available nodes, thus authors conclude that the performance is more consistent as the results show almost identical because the queue is practically null.

The CASA [8], on the other hand, is a phase-by-phase solution, which consists of two sub-scheduling algorithms; the job submission phase and the dynamic scheduling phase. Significant results from CASA include the following. The algorithm h_{assign} adopts a probabilistic approach for selecting nodes without greedily selecting the best ones which will lead to starvation. The decentralized scenario with global knowledge based on the global resource discovery service allows nodes to find other remote nodes during the job submission phase. In addition, centralized and decentralized scenarios share the same average resource uptime, thus decentralized and dynamic CASA can really promote a resource usage level similar to the BestFit heuristic algorithm. Moreover, centralized static algorithms request resource information in contrast with CASA that doesn't request either information or control authorities data, thus offering great flexibility. The decentralized with partial knowledge scenario with the data of only 6 nodes, offer the most optimized job response time of all scenarios. The results show that this solution is far better than the independent and centralized scenarios. The simulation result shows that job slowdown, waiting times, response times and messages overhead values have been significantly improved.

Having said that, the essential characteristics of previous works could lead to requirements that have been addressed from the aforementioned discussion summarized as follows:

- *Overall view*: Most of the existing works [3][4][10] aim to optimize job scheduling from the scope of improving individual participants' performance of grids and clouds without considering an overall performance view as [ye]. For providing an aggregated view of inter-cloud workloads it is required that the meta-scheduling will aim to an overall performance improvement rather than the individual nodes, or clouds.
- *Knowledge level*: Most of the existing works as in [4][5][11] assume that the (meta-) schedulers have a complete knowledge of the entire pool of resources at any given time. In practise a partial knowledge of a specific pool of participants is more realistic and could offer more flexibility to the environment.
- *Processing Information*: Most of the existing works as in [6][11][12] require detailed real time information processing which involves possibly latencies due to the data exchanging and authorization procedures of the remote nodes. However, as have been proven by [8] real time processing of nodes data could offer significant advantage to the meta-scheduling process.
- *Real Time Responses*: Most of the existing works as in [5][9] make job allocation decision prior to the job scheduling phase, without considering real-time responses. Equally, the management of unpredictability is low. On the other hand, in the case of remote nodes real time response there is no need to expose internal information during cooperation e.g. scheduling queues.
- *Rescheduling*: Most of the existing works as in [3][4][5] don't consider the concept of rescheduling in order to adapt to the unpredictability of the resources. Using re-scheduling, great flexibility can be achieved as well improvements of the initial scheduling decision.
- *History records*: Most of the existing works as in [5][7] don't consider the deployment of history records with the aim of improving re-scheduling. It has been proven by [8][13] that the usage of recorded previous work delegations during meta-scheduling can improve performance significantly.
- *Transparent user mappings*: Most of the existing works as in [4][6] don't consider mapping user request transparently.
- *Fully decentralization*: Decentralization is common to most of the works as it offers great flexibility and scalability.

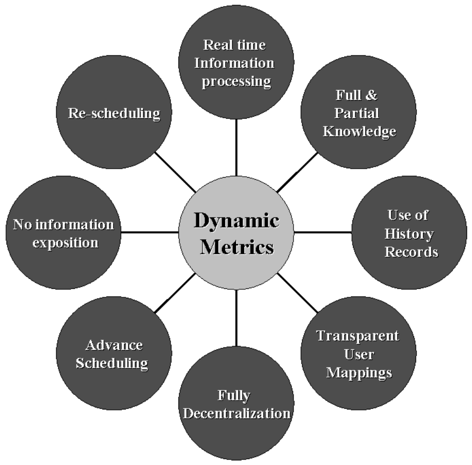

Figure 3: Dynamic Metrics for Inter-Clouds Meta-Scheduling

Thus we have concluded that the essential dynamic requirements are extracted from [13][8] could lead to the development of meta-schedulers to be adoptable in inter-clouds. Figure 3 demonstrates the dynamic features when modelling the requirements for inter-clouds.

In particular, we suggest that the dynamic metrics should support a real time processing of information environment for controlling dynamic unforeseen situations. In addition, we suggest communication without information exposition, for avoiding authorities' control and use of history records in re-scheduling or advance scheduling procedure. Finally, testing and comparison of benchmark results should be presented in full and partial knowledge scenarios in which transparent user mapping is deployed in a decentralized environment.

VI. Conclusion

The work here aims to model the requirements for inter-cloud meta-schedulers. We have described two approaches that their characteristics are mapping to the inter-cloud requirements. Thus, through a detailed discussion of their functionality, we have addressed the required dynamics when developing inter-cloud meta-schedulers.

The future prospect of the work includes the development of a meta-scheduler for inter-clouds. In addition, this effort will continue the works of [13][8] for achieving job scheduling across independent resources in distinct administrative domains. It should be mentioned, that energy efficiency (Green Clouds) will be considered. This can be achieved by incorporating migration mechanisms, during re-scheduling; thus allowing transferring workloads from high energy consuming clouds to low ones as one of the future meta-algorithm evaluation criteria.

Acknowledgements

The first author would like to thank UNITE FP7-248583 for their secondment support.

VII. References

[1] 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

[2] Christodoulopoulos, K., Sourlas, V., Mpakolas, I., and Varvarigos, E., A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in Grid networks, Computer Communications, Volume 32, Issues 7-10, 28 May 2009, Pages 1172-1184, ISSN 0140-3664

[3] De Assuncao, M., and Buyya, R., Performance analysis of allocation policies for interGrid resource provisioning. Information and Software Technology, 51(1):42{55, 2009

[4] De Assuncao, M., Buyya, R., and Venugopal, S., InterGrid: A case for internetworking islands of Grids, Concurrency and Computation: Practice and Experience 20 (8) (2008) 997–1024.

[5] Folling, A., Grimme, C., Lepping, J., and Papaspyrou, A., Decentralized grid scheduling with evolutionary fuzzy systems. In Job Scheduling Strategies for Parallel Processing, pages 16{36. Springer, 2009

[6] Frerot, C. D., Lacroix, M., and Guyennet, H. (2000a). Federation of resource traders in objects-oriented distributed systems. (PARELEC'00) August 27 - 30, Quebec, Canada.

[7] GICTF White Paper , 2010 Global Inter-Cloud Technology Forum, Use Cases and Functional Requirements for Inter-Cloud Computing, August 9, 2010, Available at: http://www.gictf.jp/doc/GICTF_Whitepaper_20100809.pdf, accessed at: 02/07/2011

[8] Huang, Y., Bessis, N., Norrington, P., Kuonen, P., and Hirsbrunner, B., Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm, Future Generation Computer Systems, In Press, Accepted Manuscript, Available online 13 May 2011, ISSN 0167-739X

[9] Xu, B., Zhao, C., Hu, E., Hu, B., Job scheduling algorithm based on Berger model in cloud environment, Advances in Engineering Software, Volume 42, Issue 7, July 2011, Pages 419-425, ISSN 0965-9978

[10] Istin, M., Visan, A., Pop, F., Dobre, C., and Cristea, V. 2010. Near-Optimal Scheduling Based on Immune Algorithms in Distributed Environments. In Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '10). IEEE Computer Society, Washington, DC, USA, 439-444.

[11] Korkhov, V. V., Moscicki, J. T., Krzhizhanovskaya, V., V., Dynamic workload balancing of parallel applications with user-level scheduling on the Grid, Future Generation Computer Systems, Volume 25, Issue 1, January 2009, Pages 28-34, ISSN 0167-739X

[12] Lai, K., Huberman, B. A., and Fine, L. (2004). Tycoon: an Implementation of a Distributed market-based resource allocation system. Technical Report, HP Labs.

[13] Leal, K., Huedo, E., and Llorente, I.M., A decentralized model for scheduling independent tasks in federated grids. Future Generation Computer Systems, 25(8):840-852, 2009. 21, 27

[14] Pop, F., Dobre, C., Stratan, C., Costan, A., and Cristea, V. 2010. Dynamic meta-scheduling architecture based on monitoring in distributed systems. Int. J. Autonomic Comput. 1, 4 (December 2010), 328-349.

[15] Sotiriadis, S., Bessis, N., and Antonopoulos, N., Towards inter-cloud schedulers: A survey of meta-scheduling approaches, Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Barcelona, Spain, Oct 26-28 2011 {To appear}

[16] Weissman, J. B. and Grimshaw, A. (1996). Federated model for scheduling in widearea systems. HPDC'96: Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, pages542-550, August

[17] Xhafa, F., and Abraham, A., Computational models and heuristic methods for Grid scheduling problems, Future Generation Computer Systems, Volume 26, Issue 4, April 2010, Pages 608-621, ISSN 0167-739X