Critical Friend Model: A Vision Towards Inter-cooperative Grid Communities

Ye Huang*, Nik Bessis†, Amos Brocco*, Pierre Kuonen‡ and Beat Hirsbrunner*
*Department of Informatics, University of Fribourg, Switzerland
Email: {ye.huang, amos.brocco, beat.hirsbrunner}@unifr.ch
†Department of Computer Science and Technology, University of Bedfordshire, UK
Email: nik.bessis@beds.ac.uk
‡Department of Information and Communication Technologies,
University of Applied Sciences Western Switzerland
Email: pierre.kuonen@hefr.ch

***Abstract*—Much work is under way within the distributed computing community in order to assign a job to an appropriate resource discovered from a fully decentralized and heterogeneous infrastructure with reasonable cost, such as optimized job responsible time, executing price, etc. However, local resources of individual virtual organizations (VOs) are managed under independent policies and constraints, therefore existing solutions are normally designed for specific scenarios and lack of commonality. In addition, boundaries of different VOs raise extra difficulties on job sharing and collaboration amongst distributed nodes. On the other hand, the obtained knowledge from multi-node cooperations is normally discarded, although it in future may lead to intelligent scheduling decision by means of previous collaboration records and experience. Especially, the trust built up according to historical collaboration between nodes from different VOs may overweigh and cross the boundaries of VOs themselves. In this work, the Critical Friend Model (CFM) grid scheduling solution is proposed to bridge the aforementioned gap between decentralized nodes and VOs. The Critical Friend Model is comprised of a set of general workflows and algorithms to make better use of node's knowledge of the neighborhood derived from historical collaboration, which is kept in the local storage and known as the metadata snapshot. In addition, a set of related grid components are also introduced to give the visible implementation roadmap of the CFM in the near future.**

***Index Terms*—Inter-cooperative grid architecture; Critical Friend Model (CFM); Metadata snapshots; SmartGRID; Community-Aware Scheduling Protocol (CASP).**

I. Introduction

Job sharing between decentralized distributed heterogenous nodes has been the goal of distributed computing both in academic and industrial fields for decades. The difficulties exist not only because the technical complexity of inter-operation betweens nodes, but also policy constraints and boundaries between various institutes and virtual organizations (VOs). In this case, an effective scheduling approach which aims at being able to fit scaled resource community needs to get across the boundaries of different VOs by utilizing unexploited information such as historical collaboration credits and knowledge [1].

The idea of this paper is to use heuristic data to facilitate the scheduling decision making process. Especially, exploiting historical interoperation metadata cached on each grid node would lead to a *demand centered* grid scheduling framework across multiple VOs. As mentioned above, conventional grid VOs are bounded due to various non-common reasons, and realistic job delegations normally only happen between nodes within the same VO. Such approach does not take the full advantage of the fact that a node could belong to more than one VO; especially when job delegation to a node of another VO will not result in a conflict to the original purpose, e.g., critical security issue that only allows job execution within the same VO.

The main contribution of this work is the proposal of a novel scheduling solution named the Critical Friend Model (CFM), which make use of the metadata of each participating node, i.e. the knowledge of neighborhood grid topology and records of previous interaction (either direct or indirect), to generate an empirical scheduling decision across individual nodes from independent VOs. The CFM considers interconnected nodes known via historical realistic collaboration records as the Self-led Critical Friends (SCF) to each other, and together represent a Critical Friendship based Community that has crossed the boundaries of isolated VOs. Furthermore, the strength of the Critical Friendship between nodes is determined by more "subjective and empirical" factors, such as the quantity and quality of previous interactions.

Nodes adopting the Critical Friend Model can work together well since they are following the same philosophy; moreover, to maximize the effectiveness and enable collaboration with other nodes that follow different scheduling solutions and philosophies, the CFM is designed to be easily integrated with a general scheduling guideline entitled Community-Aware Scheduling Protocol (CASP) [2], which has proven to be capable of providing adaptive and effective scheduling solutions in dynamic network [3].

Besides the theoretical model, a set of practical grid components are also considered as important with regard to their help towards the implementation of a Critical Friend Model based distributed computing infrastructure. In this case, an existing project named the SmartGRID is targeted for the future implementation.

The SmartGRID is a cooperative project aiming at increasing the efficiency, robustness, and reliability of heterogeneous

grid computing infrastructures [4] concerning volatile and dynamic resources. The proposed grid middleware has been designed as a generic and modular framework supporting intelligent and interoperable grid resource management using swarm intelligence algorithms and multi-type grid scheduling. SmartGRID uses a layered architecture and aims at filling the gap between grid applications, which act as the resource consumers, and the grid resource low-level management systems, which behave as the resource providers. To achieve this goal, SmartGRID uses an autonomic and evolutional grid community composed of its grid schedulers called MaGates [5].

Within the SmartGRID, the discovered information for each specific task is currently discarded after its usage. We aim at extending this model so that each node of a SmartGRID community might also be capable of keeping a metadata snapshot of known remote nodes, in order to facilitate a more efficient and intelligent behavior towards relevant scheduling decisions. Moreover, as the SmartGRID architecture strives to provide intelligent scheduling for the scope of serving the grid community as a whole, not just for a single grid node, the extended work is also concerned with the design of a scheduling strategy supporting the combination of various interoperable bounded grid communities. In this case, we propose to exploit already discovered grid nodes and store metadata snapshots that would facilitate more convenient, efficient and intelligent subsequent resource discovery operations.

The remainder of the paper is organized as follows: related knowledge concerning the trust and correlation on distributed computing is introduced in the next section. The strategy and principle of this work is introduced in Section III firstly, and then detailed in Section IV. A scenario case study is illustrated in Section V, and the work is summarized in Section VI.

II. Trust Related Overview

The Critical Friend Model is established upon the conception of trust in computer sciences.

A. Trust in Computational Environment

[6] defines trust (or symmetrically, distrust) as a particular level of subjective probability with which an agent assesses and monitors that another agent or group of agents is capable to perform and deliver a particular action. The notion of trust as an expectation (as a rational, affective or a mixture of both) is also a closely related concept to that of confidence levels. The idea that the confidence level can be measured is developed by [7] who pointed out that we arrive at the concept of trust by choosing to put ourselves in someone else hands, in that the behaviour of the other determines what we get out of the situation. Others [8] [9] have also stressed the importance of trust building over time (i.e. the temporal dimension of trust). Trust is not only clearly dependant on our past experiences but is also an expectation of reliability and confidence in future events too. Work relevant to the reputation notion has already been carried out by researchers in developing and applying various mathematical formalisms that can be used to design and implement trust models embedded within autonomic systems. Many of them rely on the calculation of local trust thresholds of various kinds. This approach has also been extended so as to seek to enable MAS (Multi-Agent Systems) with the power to investigate trust credentials, provenance and reputation. Confidence levels can also be calculated in various ways. Within the specific context of the consumption and provision of Grid services from VOs previously published work has shown the value of computationally heavyweight confidence engines as exemplified in [10]. There are also works as described in [11] which incorporate high-level proxy measures of VO reputation using purely rational measures derived from previous performance history.

B. Critical Friends and Self-led Trust

The work herein is based upon the notion of critical friends [1] and self-led trust management. The main concept involved is that a community of VO users (encompassing service consumers and providers) can communicate within their own VO network, and manage their own perceptions of other users. In such a system, a user, for example, Alice, can decide (based upon limited local knowledge) how much trust to place in another user, Bob. It is also very important to emphasize that users may belong to more than one VOs and thus, the many-to-many relationship between users with users and users with VOs can further strengthen the level of trust or mistrust. The concept herein is that the VO system is fluid and dynamic, and based upon a series of interactions between users, the trust value between users can evolve over time. The notion of critical friends and self-led trust mirrors the notion of trust relationships in the real world. If a person does not know someone or something, they will ask their friends about it. Based upon the feedback they receive a judgement (personal, and self-led) can be made. Meanwhile, a weight filter mechanism can be introduced upon the degree of usefulness of opinions previously provided by the Critical Friend. Our notion moves away from a centralised trust authority as this could lead to a single point of failure.

III. Strategy and Principle

As originally proposed in [1] and then further discussed in [12], in order to construct a collaborative computing environment across multi-VOs, we propose a novel model entitled *Critical Friend Model (CFM)*. The basic idea of the CFM is to utilize historical collaboration records, which are considered as trust correlation between nodes according to previous experience, to facilitate the job sharing and node cooperation regardless the boundaries of various Virtual Organizations (VOs). Within the Critical Friend Model, the notion of the *Self-led Critical Friend* is introduced as a mean of describing the interaction between nodes in a wider, larger- scale and unknown grid community.

The CFM is partially inspired by an existing grid project named the SmartGRID. The SmartGRID offers a loosely decoupled grid architecture in order to ensure the grid scheduling activities is independent from specific adopted resource discovery services. Two approaches are currently available for

job delegation on appropriate remote nodes, i.e. the neighboring nodes list based local search policy and the on-demand community search policy.

However, due to the original targeted scenario of the SmartGRID, the node negotiation and cooperation activities are limited within specific community scope because of nodes of the SmartGRID has to adopted the same Ant-based resource discovery service [4]. It doesn't make good use of the fact that a SmartGRID node could be a member of another VO, e.g., another information system, so that gaps amongst diverse VOs can be bridged somehow. It is then one of contribution of this paper to broaden the node cooperation topology of single VO into multi-VOs. The obtained novel topology comprised of several VOs is entitled as the *Critical Friend Community (CFC)*.

Moreover, due to the knowledge of reachable remote nodes on single participating node is enlarged, job scheduling decisions making process is supposed to be improved because more information from different sources can be obtained compared to the conventional grid, as well as more tasks have to be disposed. In this case, the CFM is comprised of a set of general patterns and algorithms, which are used to organize resources and tasks from both local users and remote collaborators according to specific user preference.

Another noticed defect of traditional usage is that the discovered information for specific task is discarded after usage, although it contains more information rather than a single action, e.g., the trust weight between two independent grid nodes. To overcome such weakness and to make better use of the archived historical information, particular attention is given to maintain aforementioned data and to construct a storage of the so called *Metadata Snapshot* for each involving node. The Metadata Snapshot is a set of collected and evaluated data from various sources, such as resource profile, resource status, and collaboration record archive. By considering such information, future empirical scheduling is believed being able to offer more intelligent decisions due the awareness of the context and kept up-to-date knowledge.

Finally, to enable the cooperation between nodes no matter whether they follow the philosophy of the CFM or not, a high-level scheduling protocol entitled *Community-Aware Scheduling Protocol (CASP)* [2] is to be integrated so that the CFM is able to work together with other scheduling policies despite the extra- facilities.

IV. Implementation

The concerning components of the Critical Friend Model which are introduced in previous section will be detailed as follows:

A. SmartGRID framework

Currently, the SmartGRID architecture consists of three parts: the Smart Resource Management Layer (SRML), the Smart Signaling Layer (SSL), and the Data Warehouse Interface (DWI).

The topology of the SmartGRID is an interoperable grid scheduler community composed of engaged decentralized grid schedulers from the SRML, named as MaGates (Magnetic Gateway) [5] and designed to be modular and emphasizing scheduler interoperation. With the infrastructure information retrieved from the DWI, the MaGates discover and connect to each other, so as to collaborate in order to bridge heterogeneous grid systems with a consensual view. The grid community evolves dynamically, and is able to automatically recover from failure situations. Information about available resources and network status is gathered by the SSL, and stored into DWI's distributed data storages. The SSL maintains an overlay network of Nests that provide the runtime environment for the execution of bio-inspired ant algorithms [13] [14]. This approach provides an adaptive and robust signaling mechanism, supporting both grid resource discovery as well as monitoring. The layered architecture of the SmartGRID is shown in Figure 1.

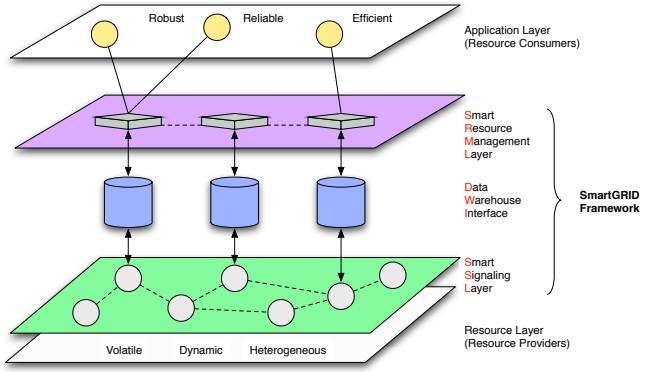

Fig. 1. SmartGRID architecture overview

The SRML is responsible for grid level dynamic scheduling and interoperation that provides grid applications with scheduling decisions on dynamic discovered computing resources. Being the core of SRML, the MaGate is also in charge of propagating resource discovery related tasks to the SSL, analyze the returned results, and decide future operations.

In general, the mission of a grid scheduler is to discover appropriate resources for executing jobs across within a single grid community. The vision of the MaGate scheduler is that of a wider grid community scheduling process is able to exploit resources in large and partially unknown grid communities, and dealing with continuously changing job queues. Thus, since each community node is supposed to receive jobs from both its local and remote grid communities, management of the job queue must deal with a more dynamic, fluid and unexpected environment. The goal is to ensure robustness, reliability, efficiency, and intelligent scheduling response. In this respect, the scheduler must compromise between accepting community jobs and local ones, depending on its workload, *agreement offers* [15], and the ratio between resources contributed to the local and the global grid communities.

The SSL [4] represents the interface from and to the network of the SmartGRID architecture, by providing access to Virtual Organization (VO) resource. The SSL is controlled by the SRML, and provides information about the availability of other resources on other nodes, as well as their status.

From the SSL point of view, each node has some partial knowledge of the underlying logical network. Remote nodes that fall into this partial view are called *direct neighbors*, because they are considered as having good connection with the host node. The SSL hides the complexity and instability of the underlying network by offering reliable services based on distributed ant algorithms. Ant algorithms do not require centralized control, and are known to be robust and adaptable, thus well suited for dynamic networks. Ants are defined as lightweight mobile agents traveling across the network, collecting information on each visited node: a distributed middleware named Solenopsis [16] provides an environment for the execution of ant colony algorithms, in particular the specific designed BlåtAnt collaborative ant algorithm [17]. The activity of the SSL can be either reactive or proactive. Reactive behaviors are controlled by incoming requests from the SRML: information is asynchronously transmitted through a data warehouse interface, and fetched by the local nest. The same interface is used to provide feedback and results on the execution of algorithms. Continuous pro-active activities, such as network monitoring, are used to enhance the QoS of provided services for the SRML, and the robustness of the whole system.

B. Extended Topology

As mentioned before, the current SmartGRID network topology implies that propagated ants searching resources within a specific grid community, which is bounded due to various reasons, such as shared community policy, trust issues, geographical location, etc. Let us now label this bounded grid community as a Virtual Organization 1 (VO_1). In a similar vain, let us assume that there are a number of separated VOs across a wider (larger-scale and thus unknown) grid community (VO_1, ... VO_n). Let us also assume that an individual node is member in more than one VO (e.g., VO_1, VO_2) and that each node within a VO can be a service consumer, service provider or both.

This extended inter-cooperative grid vision, entitled as Critical Friend Community (CFC) and illustrated in Figure 2, enables a network which clearly extends the aforementioned SmartGRID topology. This is mainly due to the fact that the current SmartGRID framework takes job delegation decisions on the basis of ants searching across one and only one VO (e.g., VO_1) and it does not take the full advantage of the fact that a node in a VO_1 can be also a node in a VO_2. In such a grid community, a node, for example, n_1 in VO_1 can communicate with another neighbor node, n_5 in the same VO_1. The rationale is that a node, n_5 can be also a member of another VO_2, which in turn leads to the rationale that n_5 can communicate with another neighbor node such as n_9 (that is also a member of VO_2 and VO_3) and so on. The assumption here is that communicating and/or delegating a job to a node belonging to a different VO will not result to a conflict of interest between parties. Having said that, the assumption is valid given the fact that a VO should not allow membership of a distinct node been part of two conflicting VOs unless it is unknown or there is a certain level of trust. In the case of the latter point, the assumption is still valid given the fact that the associated *agreement offer* and policies explicitly specify the range of act of a job delegation (what is acceptable). On the other hand, if decisions made by two interacting nodes conflict, an agreement [15] based negotiation mechanism [18] can be introduced to address such issues.

The vision is also based upon the very important notion of the *Self-led Critical Friends* (SCF). This concept is built upon relations between nodes, and the knowledge that each node constructs about some neighbor node (either a member of the same or of a different VO) based on previous (direct and/or indirect) interactions, such as communications and delegations. The notion of previous interactions between neighbor nodes determines the strength of the relation and ultimately the level of Critical Friendship. We thus consider a topology of a wider dynamic grid community, based upon a series of SCF relations between nodes from different communities; the strength of a relation between two nodes can be either constant or can evolve over time and influence decisions of the job delegation task. This idea is similar to the one proposed in [19], and effectively creates a second level overlay that can be exploited for efficient resource discovery.

The concept of SCF mirrors the notion of relationships occurring in the real world. If a person (node in our case) is looking for a specific service and they do not know how to find it, they will ask some of their friends (neighbor nodes) who may know it (decision based on past experience). If they do not, these friends will pass the query on to their own friends with the view that someone across the "friends" network (neighbors network in our case) will know and have information relevant to the original request about the specific service. Based upon this information a decision can be made. We purposely moved away from a centralized authority as this could lead to a single point of failure. To explain further, a centralized authority could be compromised by an external entity, and if all users are dependent upon this entity then the functionality of the whole network could fail very quickly. This extended resource discovery topology clearly increases scalability and thus is the most suitable solution for wide grid communities.

C. Local Policies of the Critical Friend Model

The Critical Friend Model (CFM) is able to make decisions based on information from both infrastructure providers and knowledge of critical friends. The CFM is supposed to be understood and carried out by a coordinator component. Taking into consideration that each node within the Critical Friend Community (CFC) has its own local scheduling policies, as well as full control of the local resources, the coordinator is supposed to collaborate with the existing local scheduling policies, and provides a broad view by enabling the participation of remote nodes.

A coordinator is different from a meta-scheduler, although they may be physically the same component sometimes. A meta-scheduler simply assigns a job to local LRM for execution, while two coordinators have to negotiate on a job delegation from one node to the other. That is to say that a

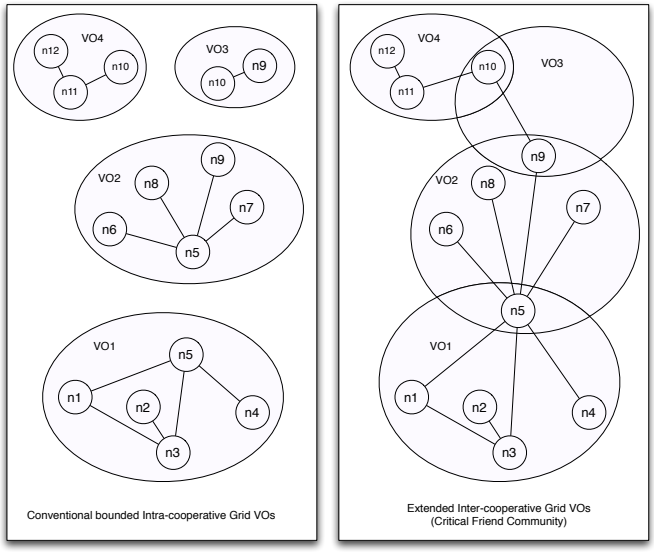

Fig. 2. Vision of Intra-cooperative and Inter-cooperative Grid Topologies (Critical Friend Community)

job delegation request issued by a coordinator can be refused or altered, which is not the case for a meta-scheduler.

If a job delegation request is refused or altered, the initiators coordinator has to continue by either reporting the failure to user, or releasing a re-negotiation process with modified parameters. The approach to automate the above process can be comprehended as a *workflow based* schedule, because the coordinator has already determined steps to do for handling subsequent behaviors like re-negotiation and failure. Regarding the scheduling process of each CFC node concerns many volatile factors retrieved from various environments, adaptability is a critical capability to exploit potential opportunities of fulfilling received job execution requests without bothering the initiator user.

The Critical Friend Model is comprised of two behavior patterns, namely the *Job Arrival Pattern* and the *Job Complete Pattern*.

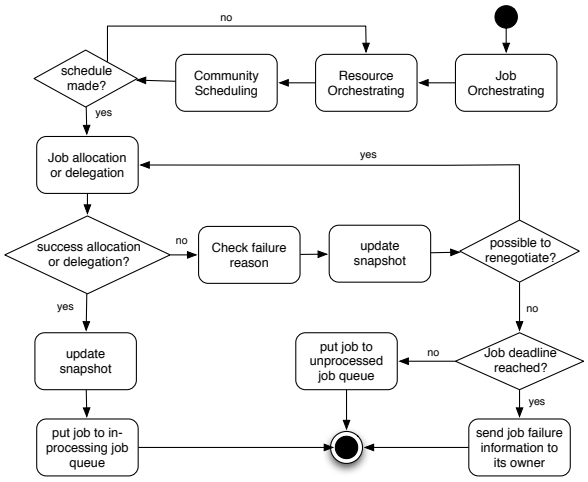

Fig. 3. Critical Friend Model Job Arrival Pattern

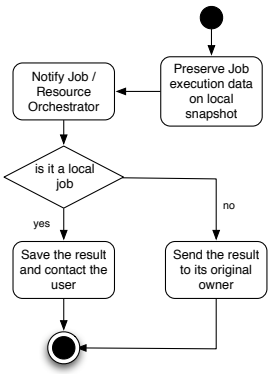

Fig. 4. Critical Friend Model Job Complete Pattern

The Job Arrival Pattern, as illustrated in Figure 3, starts from the *Job Orchestrator*, a component responsible for generating next to-process job depending on continuously arriving incoming jobs, no matter where they come from (either the local node or a remote one). Conversely, the Job Complete Pattern, as illustrated in Figure 4, is triggered by notification events of newly accomplished jobs, from either local resource or remote nodes, in order to preserve useful data retrieved from job execution (e.g., job SLA satisfaction [20], used resource CF weight re-calculation, etc.) for future use.

Besides, the CFM is supposed to be complemented by several local policy relevant orchestration algorithms, such as the simplified Job Orchestrating Algorithm, Resource Orchestrating Algorithm, and Community Scheduling Algorithm, which will be discussed below.

An interaction mechanism between involving coordinators is a critical component for the design of the CFM. Such a mechanism should be flexible to adapt to different scenarios, platform independent to decouple the participators from the infrastructure, robust to recover from an unavailable/failed agreement, and automated to handle continuous incoming requests.

More specifically, the implementations of the aforementioned algorithms are detailed as follows:

1) Job Orchestrating Algorithm (JOA): The philosophy of JOA is to organize a to-process job queue by merging diverse job incoming sources, with respect to local user preference.

If the preference indicates that local jobs have higher priority, the JOA will try to fill the size limited output queue with jobs from local queue firstly, and pick appropriate jobs from other sources, e.g., community queue or unprocessed queue, only if the limit of the output queue is not exceeded. If the user desires an equal treatment for all incoming job requests, the output queue will be comprised of the earliest arrived jobs, no matter where they come from. Finally, if a profitable philosophy is determined, each arrived job will be evaluated, in order to determine individual *job-profite-rate* value. In this case, the output queue will be composed by the most profitable jobs. Once the output queue is generated, the local policy of the participating node is invoked for future processing.

Furthermore, the JOA can be extended by users self-defined

job orchestrating policies, and other locally adopted scheduling algorithms, besides herein mentioned FCFS and EasyBackfilling.

Algorithm 1 Job Orchestrating

Require: a local job queue lj, a community job queue cj, a unprocessed job queue uj, a output job queue oj, allowed size of oj $limitoj$, current size of oj $sizeoj$

Require: $jpr = jp/(jtc * jcpu)$
jpr: job profit rate; jp: job profit
jtc: time cost of a job; $jcpu$: required number of cpu of a job

1: $local.scheduler \in \{FCFS, EasyBackfilling\}$
2: $user.preference \in \{LocalJobPriority, CommunityJobFair, Profitable\}$
3: update data of lj, cj, uj
4: **if** $user.preference = LocalJobPriority$ **then**
5: fill oj with $(limitoj)$ jobs from lj
6: **if** $limitoj$ is not reached **then**
7: find $(limitoj - sizeoj)$ jobs from cj and uj to fill oj
8: **end if**
9: **else if** $user.preference = CommunityJobFair$ **then**
10: fill oj by selecting $(limitoj)$ jobs from lj, cj and uj depending on arrival time fairly
11: **else if** $user.preference = Profitable$ **then**
12: **for** job from queue lj, cj, uj **do**
13: **if** jpr of job is not determined **then**
14: calculate jpr for job
15: **end if**
16: **end for**
17: fill oj by selecting $(limitoj)$ jobs from lj, cj and uj, depending on jpr fairly
18: **else**
19: generate oj with random picked jobs
20: **end if**
21: determine the validate time of oj
22: output oj
23: invoke $local.scheduler$ to execute oj

2) Resource Orchestrating Algorithm (ROA): The ROA is responsible for generating a set of appropriate candidate resources for each input job request, depending on user preference.

If the *LocalResourcePriority* policy is chosen, resources owned by the local node are considered firstly, with an additional selection from other list (e.g., community resource list and critical friend resource list) only occurring if the size limit of the output resource list is not achieved. If the policy *CommunityResourceFair* is preferred, a fair selection is carried out on all known resource list. Finally, if the policy *FriendResourcePrioriy* is specified, the output list will firstly pick up a suitable resource owned either locally or by some critical friends, with other list not being considered unless the output resource list is not full.

Similarly to the aforementioned JOA, the ROA can be extended by user self-defined resource orchestration policies, but only if the expected known resource list can be found within the local node's snapshot storage.

Algorithm 2 Resource Orchestrating

Require: a local resource list lr, a community resource list cr, a critical friend resource list fr, a output resource list or, limit of output resource list $limitor$

Require: incoming job requirement jr

1: $user.preference \in \{LocalResourcePriority, CommunityResourceFair, FriendResourcePriority\}$
2: update data of lr, cr, fr
3: **if** $user.preference = LocalResourcePriority$ **then**
4: fill or with selected resources from lr firstly
5: **if** $limitor$ is not reach **then**
6: fill or with fairly selected resources from cr and fr
7: **end if**
8: **else if** $user.preference = CommunityResourceFair$ **then**
9: fill or with fairly selected resources from lr, cr and fr
10: **else if** $user.preference = FriendResourcePriority$ **then**
11: fill or with fairly selected resources from lr, fr
12: **if** $limitor$ is not reach **then**
13: fill or with selected resource from cr
14: **end if**
15: **else**
16: generate or with random selected appropriate resources
17: **end if**
18: output or

3) Community Scheduling Algorithm (CSA): Once a candidate schedule (a job with its candidate resource list) arrives, an allowed maximum scheduling time duration will be given to prevent unacceptable delays and performance loss. The CSA is responsible for contacting the candidate resources simultaneously within allowed delay, in order to get a job allocation/delegation *agreement* [15] based on the expected request (in our case, it is an *agreement offer*). An *agreement* means that the job execution request is approved by the target resource (either locally or remotely) and if such job can be delivered within a certain time, it will be accepted and executed under the agreed terms. As soon as an *agreement* has been made between the requesting node and a target resource, other *agreement offers* will be revoked.

In case no candidate resources are able to accept such *agreement offer* due to various reasons, e.g., local workload, local policy alternation, latest resource status change, the CSA needs to check whether the allocated scheduling time has expired. If not, the CSA is able to contact the locally adopted Information System, and asks for a live search from the located VO within the remaining scheduling duration. If appropriate resources can be found within such time constraints, a parallel (re-)negotiation with a newly prepared *agreement offer* can be issued again, within the shortened time duration.

As mentioned, although the job allocation is a different operation from job delegation (because the targeted resource of job allocation is an owned LRM of the local node, which

cannot negotiate a job acceptance), the CSA is not concerned with such slight difference and ignores the *agreement offer* based (re-)negotiation process if the target resource is managed by a local LRM.

Algorithm 3 Community Scheduling Algorithm

Require: prepared candidate resource list cr
Information System of the located VO IS
current processing job job
allowed scheduling time ts, current time t, expected time deadline $tstop = (t + ts)$
agreement offer on job allocation/delegation $offer$
agreement on job allocation/delegation $agreement$

```
get the allowed scheduling time ts for job
repeat {parellel}
    negotiate/re-negotiate offer within ts
    if agreement made then
        break
    end if
until each resource of cr is contacted
if agreement not available then
    if t < tstop then
        invoke IS within (tstop − t)
        repeat {parellel}
            negotiate/re-negotiate offer within (tstop − t)
            if agreement made then
                break
            end if
        until each discovered result of IS is contacted
    end if
end if
if agreement not broken and agreement not expired then
    allocate/delegate job based on agreement
end if
```

D. Metadata Snapshots

Metadata Snapshots help to address the issue of leading to intelligent and empirical future scheduling decisions according to previous job sharing experience.

As mentioned earlier, every node within the grid community must publish its capabilities; moreover, the assumption is that this public profile is kept continuously up-to-date. Specifically, within the framework of SmartGRID, each MaGate scheduler generates directives to propagate ants in the SSL in order to provide nodes with availability and status information. Current MaGate resource discovery service either utilizes partial knowledge stored in a node's cache, or delivers search queries to match individual job delegation requirements with published node profiles from the known bounded grid community. To achieve this, the SSL is employed to hide the complexity and instability of the underlying network by utilizing ant algorithms. These ants function as lightweight mobile agents traveling across the grid network, collecting information on each visited node.

The herein objective is to extend current SmartGRID functionality by enabling the MaGate resource discovery service to utilize more knowledge that can be made available from the visited nodes. In this respect, we propose that each time a node, n_1, delegates a job to another node (with no regard to the possibility that it might be a member in multiple VOs) such as node n_3, node n_1 is required to keep an instance profile with regard to the parameters which have been used to discover it as a resource originally, as well as, the quality of service provided by node n_3. In a similar way, it is expected that every time a node, n_1, delegates a job to n_3, node n_1 has to update the profile about n_3 in its cache. We suggest that this is a bi-directional commodity and thus, we expect that node, n_3 will also keep an instance profile about n_1 in its cache. We also assume that a node n_1 stores as many profiles in its cache as the number of previously visited nodes. The vision is to enable nodes storing meaningful information that can help assist them and their critical friends at a later stage.

Apparently, these profiles residing in each cooperating node can be sustained or even evolve over time and act as the tool towards decision making for delegating a job next time. That is, a calculation as an aggregative and weighted value representing the (strength or else) critical friendship relationship between nodes that previously cooperated, could significantly improve the decision making towards a job delegation to a particular node (or cluster of nodes) that is available from a pool of discovered resources across the wider grid community. Such a notion clearly provides a richness not seen in any other resource discovery or job delegation model.

As mentioned in subsection IV-C, regarding many factors that could impact the scheduling decision, each node of the Critical Friend Community (CFC) has a metadata snapshot. Each metadata snapshot is comprised of sets of schemes. A scheme is a group of elements that is used for describing a particular resource or purpose. For example, a machine scheme is normally composed of elements such as machine architecture, operating system, number of CPU, etc. Other important schemes include: local resource profile, local resource status, agreement offer list, known Self-led Critical Friend (SCF) profile list, known SCF recent status list, historical processing records, etc. Noteworthy, data stored within each scheme is kept up-to-date over time, and is being evaluated and weighted to facilitate intelligent scheduling for the future incoming job requests.

To remedy the pain of organizing all necessary information (static and dynamic) of a Self-led Critical Friend (SCF), as well as to represent such knowledge is easy-to-understand way, the notion of *Snapshot Profile* is proposed. As shown in Figure 5, A *Snapshot Daemon* is responsible for gathering metadata distributed in different schemes, and representing the knowledge of each individual SCF in a clean and well-organized way. The Snapshot Profile concerns all valuable knowledge of a critical friend, including: CF location, configuration, static and dynamic status, installed application list, tariff, weight (as a CF of the local node), historical SLA (depending on job type) [20] [21], historical charge-load arrange (depending on job type), prerequisite (depending on job type), time-stamp (indicating until when this information

can be considered as up-to-date).

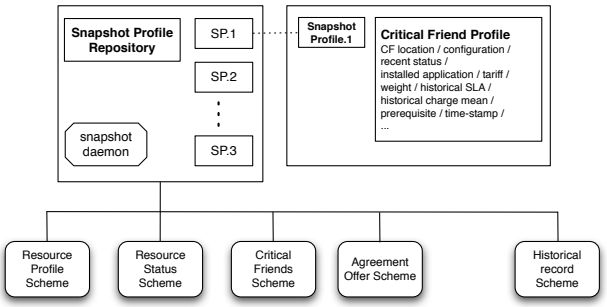

Fig. 5. Metadata Snapshot Structure

Finally, the *Snapshot Daemon* is also responsible of handling metadata exchange, either proactively or reactively, with other nodes of the CFC.

E. Community-Aware Scheduling Protocol

Besides the Critical Friend Model itself, to enable nodes following the principle of the CFM are able to collaborate with nodes with different philosophies, the integration with a general scheduling protocol, namely the Community-Aware Scheduling Protocol [2], is expected.

The basic idea of the CASP is to make scheduling decisions upon grid community, instead of isolated job queue on each grid node. The protocol dedicates to spread the all involving scheduling events, e.g., job submission and job queue optimization, across the network in order to reach as many candidate nodes as possible. The CASP is comprised of two main phases:

1) Job Submission Phase: Grid users can submit jobs to any nodes of the grid community. The initial job recipient, referred as the *initiator*, issues a resource discovery across the grid overlay by broadcasting a REQUEST message. The *initiator* then waits for a predefined timelapse for incoming query replies. A node receiving a REQUEST messages checks if the required profile matches its own capability. Accordingly, it computes an estimated completion time/cost based on actual resources and current scheduling, and forwards this information by means of an ACCEPT message.

The *initiator* then evaluates incoming ACCEPT responses, and selects the best suited node (i.e. the node providing the least time to completion or lowest execution fee). The latter is assigned the job by means of an ASSIGN message, and is referred to as the *assignee*.

2) Scheduling and Rescheduling Phase: The *assignee* is responsible to manage and execute the assigned job according to its own scheduling mechanism and policy. Based on the *initiator*'s offer selection mechanism, the *assignee* is the node that provides the shortest time-to-completion or lowest cost for that particular job. However, availability of resource on the grid and scheduling of jobs may change due to various reasons, such as new nodes connecting to the grid or existing job cancelation. Thus, the *assignee* may not remain the best solution.

In this case, the *assignee* may generate a number of INFORM messages as far as the job execution has not yet started, and send them over the network using a low-overhead random walking protocol. INFORM messages' content is similar to REQUEST messages, but their purpose is to inform other nodes about the job's current schedule on the *assignee* node. A node receiving an INFORM message checks if it matches its up-to-date profile [22] and evaluates an estimated value (e.g. completion time) according to its own schedule. If such estimation leads to a better result than the INFORM message, the node will send an ACCEPT message to the node that currently manages the job. The current *assignee* receiving the message may then choose to re-assign the job by means of an ASSIGN message.

Regarding the CASP only suggests a set of general rules towards a collaborative scheduling decision making process, it can cooperate with different local scheduling policies and mechanisms, including the Critical Friend Model.

V. Case Study

We are interested here in describing the proposed novel inter-cooperative process and related events sequence in order to illustrate the behavior of the Critical Friend Model. In the context of the SmartGRID, this involves ants as agents acting on behalf of neighboring nodes in order to enable the MaGate scheduler to discover, decide and assign job delegations on suitable resources. To achieve this, we are using the aggregative case scenario (ACS) below. Figure 6 illustrates the low-level events transferring workflow.

ACS: Let us assume that a VO_1, consists of 8 nodes (nodes n_1 ... n_8). Let us assume that a VO_2, consists of 6 nodes (nodes n_7, n_9, ... n_{13}). Let us also assume that a node, n_1 in VO_1 wishes to delegate a job to another node $n_?$, using ants a_2, a_4 and a_7, which are propagated according to queries issued by the MaGate resource discovery service. The following sequential steps describe this novel, inter-cooperative process:

1) Node n_1 in VO_1 invokes its ants a_2, a_4 and a_7;
2) Ant a_2 contacts the neighboring node n_2 (that is a critical friend: $cf_{1,2}$), ant a_4 contacts the neighboring node n_4 (that is a critical friend: $cf_{1,4}$) and ant a_7 contacts the neighboring node n_7 (that is a critical friend: $cf_{1,7}$. It's noteworthy that node n_7 is a member of both VO_1 and VO_2);
3) Ant a_2 reads and collects the public availability profile, as well as its metadata snapshots about previous job delegation activities completed in node n_2 that are available from the cache of node n_2;
4) With discovered metadata snapshots (available from the cache of node n_2) by ant a_2, MaGate n_1 realizes that node n_3 is a $cf_{3,2}$; moreover, n_3 has the capacity to take the job delegation task jd_1;
5) Ant a_4 reads and collects the public availability profile of n_4, as well as its metadata snapshots about previous job delegation activities completed in node n_4 that are available from the cache of node n_4;

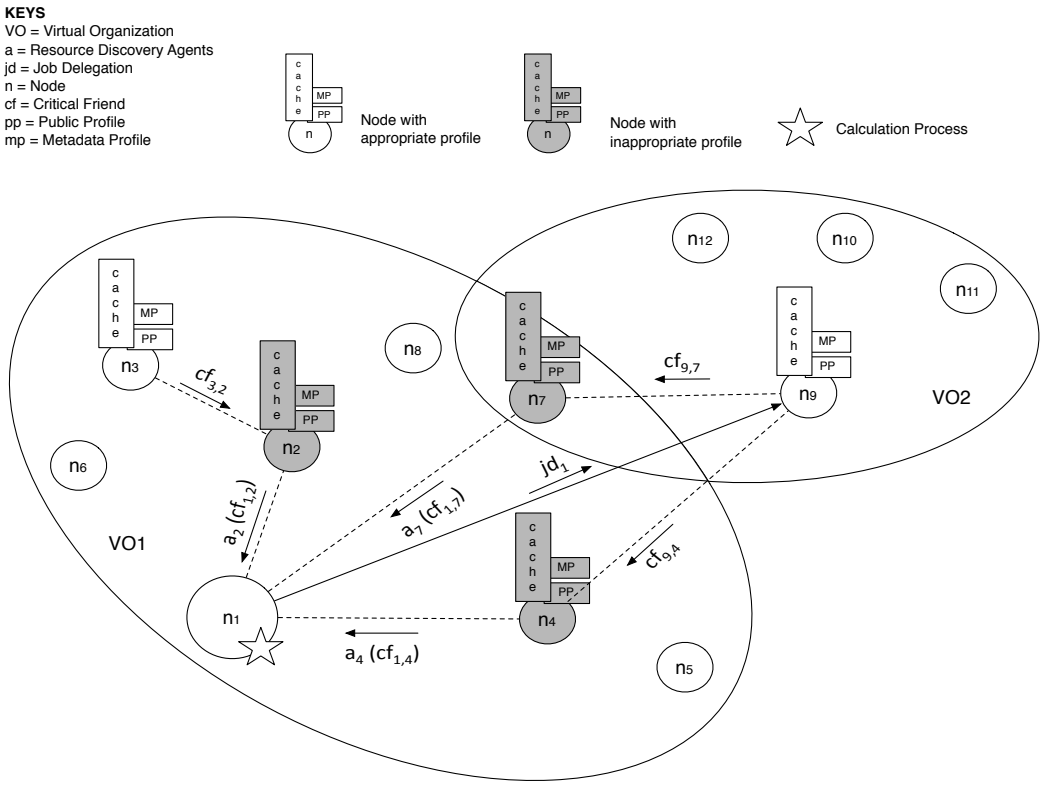

Fig. 6. SmartGRID based Critical Friend Model collaboration workflow

6) With discovered public availability profile by ant a_4, MaGate n_1 realizes that node n_4 has no capacity to take the job delegation task jd_1;
7) With discovered metadata snapshots (available from node n_4 cache) by ant a_4, MaGate n_1 realizes that node n_9 is a $cf_{9,4}$; moreover, n_9 has the capacity to take the job delegation task, jd_1;
8) Ant a_7 reads and collects the public availability profile of n_7, as well as its metadata snapshots about previous job delegation activities completed in node n_7 that are available from the cache of node n_7;
9) With discovered metadata snapshots (available from the cache of node n_7) by ant a_7, MaGate n_1 realizes that node n_9 is a $cf_{9,7}$; moreover, n_9 has the capacity to take the job delegation task jd_1;
10) Ants a_2, a_4 and a_7 collect profiles about nodes n_3 and n_9 for MaGate n_1, which reports such information to a virtualized data warehouse;
11) We now assume that a calculation as an aggregative and weighted value representing the (strength or else) critical friendship relationship between previously cooperating nodes, $cf_{3,2}$, $cf_{9,4}$ and $cf_{7,9}$ has significantly improved the decision making towards jd_1 to node n_9. This is due to the aggregative cf values that have suggested that node n_3 has been delegated x number of jobs and the satisfaction (confidence) level was significantly less than the satisfaction (confidence) level provided for an equal number of past delegated jobs in node n_9.

VI. Conclusion

In order to go beyond the boundaries of the conventional grid virtual organizations (VOs), a novel cooperative mechanism entitled the Critical Friend Model (CFM) has been proposed in this work. The kernel conception of the CFM is the Self-led Critical Friends (SCF) maintained on each participating grid node which stands for a set of remote grid nodes known by the host node and weighted according to historical collaboration records and experience. The SCF are trusted due to their previous behaviors during the collaboration with the host node, and can be contacted without regard to adopted scheduling policies, resource discovery limitation, or VO boundaries.

To demonstrate a doable roadmap, the principle, as well as theoretical components of the CFM are illustrated. The CFM presents an extended grid topology inspired by an existing grid project named the SmartGRID. The SmartGRID has been developed to be a generic and modular framework to support intelligent and interoperable grid resource management using swarm intelligence algorithms. The CFM addresses how grid schedulers from various bounded grid communities could be used in a manner that would extend current SmartGRID functionality.

In the context of the CFM, a novel cooperative mechanism that makes use of historical collaboration experience is introduced. More specifically, the CFM consists of several general patterns, e.g., the Job Arrival Pattern and the Job Complete Pattern, as well as a set of algorithms, e.g., the Job

Orchestrating Algorithm (JOA), the Resource Orchestrating Algorithm (ROA), and the Community Scheduling Algorithm (CSA), in order to serve future scheduling and collaboration behaviors well with obtained metadata within the scope of extended topology. Aforementioned patterns and algorithms can have customized preference in order to fit various local requests and constraints.

To preserve the historical collaboration records within the decentralized distributed grid nodes, a metadata snapshot is required on each participating node. The metadata on each node is established from various sources, e.g., resource profile, resource status, adopted resource discovery service cache, and node coordinator archive. Data is re-organized within the metadata snapshot into diverse Snapshot Profiles, so that they can be easily searched and analyzed.

In order to enable the interaction between nodes following the CFM philosophy and nodes which don't follow, the Community-Aware Scheduling Protocol (CASP) is proposed to be integrated. CASP is mainly comprised of two operating phases to disseminate particular important scheduling events, e.g., community job arrival, first scheduling within limited duration, and re-scheduling for nodes with long waiting time. Regarding no preference on the local facilities and local policies, CASP is able to fulfill the integration purpose and allows a transparent and non-mandatory using manner of the CFM.

The reference experiment of the CFM, as well as the complementary components, are being implemented by the responsible research groups [23] [24] [25].

ACKNOWLEDGEMENTS

This work is financially supported by the Swiss Hasler Foundation [26], in the framework of the ManCom Initiative (ManCom for Managing Complexity of Information and Communication Systems), project Nr. 2122.

REFERENCES

[1] Y. Huang, N. Bessis, A. Brocco, P. Kuonen, M. Courant, and B. Hirsbrunner. Using Metadata Snapshots for Extending Ant-based Resource Discovery Service in Inter-cooperative Grid Communities. In *International Conference on Evolving Internet*, pages 89–94, Cannes, French Riviera, France, 2009. INTERNET 2009, IEEE Computer Society.

[2] Y. Huang, A. Brocco, N. Bessis, P. Kuonen, and B. Hirsbrunner. Community-Aware Scheduling Protocol for Grids. In *24th IEEE International Conference on Advanced Information Networking and Applications*, pages 334–341, Perth, Australia, 2010. AINA 2010, IEEE.

[3] A. Brocco, A. Malatras, Y. Huang, and B. Hirsbrunner. ARiA: A Protocol for Dynamic Fully Distributed Grid Meta-Scheduling. In *30th IEEE International Conference on Distributed Computing Systems*, Genoa, Italy, 2010. ICDCS 2010, IEEE. To appear.

[4] Y. Huang, A. Brocco, P. Kuonen, M. Courant, and B. Hirsbrunner. SmartGRID: A Fully Decentralized Grid Scheduling Framework Supported by Swarm Intelligence. In *Seventh International Conference on Grid and Cooperative Computing, 2008. GCC '08*, pages 160–168, China, 2008. IEEE Computer Society.

[5] Y. Huang, A. Brocco, M. Courant, B. Hirsbrunner, and P. Kuonen. MaGate: an interoperable, decentralized and modular high-level grid scheduler. *International Journal of Distributed Systems and Technologies (IJDST)*, 2010.

[6] D. Gambetta et al. *Trust: Making and breaking cooperative relations*. Basil Blackwell New York, 1990.

[7] S.P. Marsh. *Formalising trust as a computational concept*. Citeseer, 1994.

[8] S. Dayal, H. Landesberg, and M. Zeisser. How to build trust online: Marketers can build mutually valuable relationships with customers through a trust-based collaboration process. *Marketing Management*, 8:64–66, 1999.

[9] P. Nikander and K. Karvonen. Users and trust in cyberspace. *Lecture Notes in Computer Science*, pages 24–35, 2001.

[10] M. Wilson, A. Arenas, and L. Schubert. Trustcom framework v4. *TrustCoM Deliverable D*, 63, 2006.

[11] S. Song, K. Hwang, and Y.K. Kwok. Trusted grid computing with security binding and trust integration. *Journal of Grid computing*, 3(1):53–73, 2005.

[12] Y. Huang, N. Bessis, A. Brocco, S. Sotiriadis, M. Courant, P. Kuonen, and B. Hirsbrunner. Towards an integrated vision across inter-cooperative grid virtual organizations. In *Future Generation Information Technology*, pages 120–128, Jeju Island, Korea, 2009. FGIT 2009, Springer LNCS.

[13] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, USA, 1999.

[14] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[15] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web Services Agreement Specification (WS-Agreement). Technical report, Open Grid Forum, USA, 2004.

[16] Amos Brocco, Béat Hirsbrunner, and Michèle Courant. Solenopsis: A framework for the development of ant algorithms. In *Swarm Intelligence Symposium*, pages 316–323, Honolulu, Hawaii, April 2007. SIS, IEEE.

[17] Amos Brocco, Fulvio Frapolli, and Beat Hirsbrunner. Bounded diameter overlay construction: A self organized approach. In *IEEE Swarm Intelligence Symposium*, Nashville, Tennessee, USA, April 2009. SIS 2009, IEEE.

[18] DGA Mobach, BJ Overeinder, and FMT Brazier. A WS-Agreement based resource negotiation framework for mobile agents. *Scalable Computing: Practice and Experience*, 7(1):23–36, 2006.

[19] Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient search in unstructured peer-to-peer networks. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 271–272, New York, NY, USA, 2004. ACM.

[20] A.N. Hiles. Service level agreements. *The TQM Magazine*, 6(2):14–16, 1994.

[21] R. Ranjan, A. Harwood, R. Buyya, and A. Victoria. SLA-based coordinated superscheduling scheme for computational Grids. In *IEEE International Conference on Cluster Computing (Cluster 2006), Barcelona, Spain*, pages 1–8. Citeseer, 2006.

[22] Nik Bessis. *Grid Technology for Maximizing Collaborative Decision Management and Support: Advancing Effective Virtual Organizations*, chapter Model Architecture for a User Tailored Data Push Service in Data Grids. IGI, 2009.

[23] Pervasive Artificial Intelligence Research Group, University of Fribourg, Switzerland. http://diuf.unifr.ch/pai, 2010.

[24] GridGroup, University of Applied Sciences Western Switzerland, Fribourg. http://gridgroup.hefr.ch, 2010.

[25] Centre for Research in DIstributed Technologies, University of Bedfordshire, UK. http://www.beds.ac.uk, 2010.

[26] Hasler Fundation. http://www.haslerstiftung.ch/, 2010.