UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO DINIZ MONTEIRO

IMPLEMENTING FAIR SHARING OF RESOURCES WITH FOGBOW MIDDLEWARE

CAMPINA GRANDE - PB

2019

GUSTAVO DINIZ MONTEIRO

IMPLEMENTING FAIR SHARING OF RESOURCES WITH FOGBOW MIDDLEWARE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Francisco Vilar Brasileiro.

CAMPINA GRANDE - PB

2019

M775i Monteiro, Gustavo Diniz.
Implementing fair sharing of resources with fogbow middleware. / Gustavo Diniz Monteiro. – 2019.

11 f.

Orientadores: Prof. Dr. Francisco Vilar Brasileiro.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. P2P Systems. 2. Arquitetura de redes Peer-to-peer. 3. Fairness. 4. Cloud federation. 5. Fogbow middleware. 6. FD-Nof algorithms. I. Brasileiro, Francisco Vilar. II. Título.

CDU:004.(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

GUSTAVO DINIZ MONTEIRO

IMPLEMENTING FAIR SHARING OF RESOURCES WITH FOGBOW MIDDLEWARE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Francisco Vilar Brasileiro
Orientador – UASC/CEEI/UFCG

Professor Dr. Carlos Wilson Dantas Almeida
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de julho de 2019.

CAMPINA GRANDE - PB

Implementing fair sharing of resources with Fogbow middleware

Gustavo Diniz Monteiro*
gustavo.monteiro@ccc.ufcg.edu.br
Federal University of Campina Grande
Campina Grande, Paraiba, Brazil

Francisco Brasileiro†
fubica@computacao.ufcg.edu.br
Federal University of Campina Grande
Campina Grande, Paraiba, Brazil

ABSTRACT

Nowadays most companies use cloud, and a good fraction of them use private clouds. On many occasions, these organizations have extreme fluctuations in their demand, and during these peaks, they have to resort to public clouds to meet demand. The cost of such an action in itself can be expensive, which combined with a possible underutilization of resources can generate high costs. The objective of this work is to present a tool to implement and incentive justice on resource sharing in P2P systems, initially using the direct reciprocity Network of Favors model. The direct reciprocity justice model that will be used for small and medium-sized networks, for scenarios where repeated peer interaction is most likely and where these organizations could meet their demand at peak times and offer favors in underutilization, with the guarantee of protection against uncooperative members. Also, the network focuses on being a lightweight solution, designed to be a pluggable and adaptable element to the provider or middleware used, using firsthand knowledge gained by each member through direct interaction with other peers.

KEYWORDS

federation, p2p networks, fairness

ACM Reference Format:
Gustavo Diniz Monteiro and Francisco Brasileiro. 2019. Implementing fair sharing of resources with Fogbow middleware. In *Proceedings of* . ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Organizations with varying patterns of demand and peaks of use often resort to public clouds (cf. âĂIJcloud burstingâĂIJ [10]) to meet unexpected or short-term needs, thus escaping from failures to meet business demands and service quality.

However, outside of peak hours, the resources of these organizations may become inactive, which is a loss of efficiency and a financial loss.

In this context, in the quest to maximize the efficiency of these organizations, the concept of a federation of cloud providers was created, with the purpose exchange of resources among these organizations to satisfy peak demand situations and decrease the idleness in moments of lower demand to reduce or even ending the problem.

From the architecture point of view, cloud federations can be centralized or P2P [5]. In centralized architectures, source allocation is typically performed by a trusted central entity, able to prevent free-riders(the term that we will use here to define those peers that only seek to obtain resources and not to provide them, or to provide them minimally) and achieve better matching between consumers and suppliers.

In P2P federations, on the other hand, participants must communicate and negotiate directly with one another. The advantages of decentralized topologies include extensibility, scalability, and easier deployment and management. Disadvantages, on the other hand, include difficulties in discovery, routing, security, and reliability, since the participants are largely unknown to one another and can not be considered reliable or collaborative.

Also, peers are considered selfish and have an economic incentive to become free riders, i.e. pure resource consumers.

From the economic view, the federation model can be based on payments or exchange of resources loans, which we will define here by the term 'favor', each model can be more beneficial for a certain size of the organization, your purpose and financial contribution among other aspects can also be classified as centralized or decentralized. In a centralized monetary system, bids and requests are typically collected by a central entity or market auctioneer that decides the best matching of buyers and sellers. In decentralized markets, on the other hand, buyers and sellers must explore the market by themselves and bargain directly with each other. An important question here is how efficient resource allocation can be guaranteed so that the goals of collaborative cloud members are met.

1.1 Business models for cloud federations

To better understand the pros and cons of each business model, which vary according to the context of the organization in which they would be applied these models will be described and some of their use cases exemplified.

They divide into two niches, the payment-based ones, and the non-monetary exchange-based that are so defined.

1.1.1 Pay as you go. Payment-based business model, in a postpaid form, where resources are used freely during a defined time interval and generated at the end of this payment-to-use cycle an invoice.

As a federation model, a single invoice can be generated, where the values will be distributed to the peers, or an individual invoice for each of the peers according to the use of their resources.

As a business model for cloud providers, it is used in large public clouds such as AWS, Gcloud, and Azure.

1.1.2 Credit (pre-paid). A prepaid business model where the user purchases credit from resources and makes use of the federation's resources so that usage is limited to those credits.

*Responsible for research
†Mentor

Peers providers of these resources have the values of these credits divided according to their participation in the provisioning of these resources.

This business model is commonly used in cloud providers, contracts with educational institutions or governments for example.

1.1.3 Barter. A business model based on the exchange of favors, not monetary payment, where suppliers lend their cloud resources to consumers, thus generating a debt that is also paid in the form of favors, creating a cycle of exchange of resources between peers.

In this case, each provider in the federation acts both as a resource provider (outside peak times) and as a resource consumer (during peak times). In particular private cloud providers, because of the usually limited amount of own resources, would benefit greatly from this exchange [4], thus allowing them to access a larger pool of resources when local demand is too high and can not be met using only local resources or this resources that simply do not exist locally.

In this scenario, situations of dishonesty among members are necessarily where a particular member only consumes resources from other peers and does not provide anyone or supplies in much smaller amounts that consumers can happen, situations that can lead to the abandonment of members and the collapse of the network. Control measures are then necessary to control the misuse of the network by certain members and the consequent decrease of justice in the sharing of resources.

To solve this problem, the concept of FD-Nof (Fairness Driven Network) was defined, where a model of exchange of resources is specified with justice seeking the maximization of the use of the network and internal efficiency of the pairs.

This model is an evolution of the Nof [11] model, initially described for the computational grid model and adapted to the cloud federation model, bringing a capacity control model among other improvements that will be explained later.

1.2 Implementations of Federation model

One tool used for managing cloud federations is Fogbow middleware, an open source tool that enables the integration of multiple public cloud providers, AWS and Azure for example, and private cloud middleware such as OpenStack, Cloudstack and Opennebula.

2 BACKGROUND

The purpose of this work is to create an implementation of the FD-Nof concept so that it can work for any federation provider or cloud (public or private) generically. So for this was chosen the Fogbow federation middleware, because it is open source and having support for the multiple cloud providers natively, it becomes a rather interesting choice to dock FD-Nof as default behavior so that a Fogbow driver will also be implemented and made available with FD-Nof.

Therefore, it is necessary to define the operating principles of FD-Nof and Fogbow as well as its architectures.

2.1 FD-Nof Algorithms

The FD-Nof has its operation based primarily on one metric, justice, which is defined based on the exchange of resources between the peers.

The main FD-Nof metric is justice that can be generically defined as:

$$fairness = \frac{consumed}{provided}$$

The FD-Nof calculates its justice in two ways, the first of which is the relation between what a network peer provides resources and how much it consumes from it (global justice), and the second is the relationship between consumption and how much was given between two members of the federation (pairwise justice).

$$provided == 0 \rightarrow fairness = -1$$

$$provided > 0 \rightarrow fairness = \frac{consumed}{provided}$$

The values of pairwise justice and global justice have different importance so that in an interaction between peers global justice will be evaluated only if the pairwise justice value is -1.

The great differential of the FD-Nof is the control of justice and the exclusion of the free riders of the system, it does this through a capacity controller [6] whose operation is defined below.

Algorithm 1 Capacity controller algorithm for peer A at the beginning of time step t, t > 0, to the set of other peers F and C as the maximum quota value.

```
increase = false
for all p ∈ F do
  if fairness(A, p) > 0 then
    if fairness(A, p) < TMin then
      increase = false
    else if fairness(A, p) > TMax then
      increase = true
    else if fairness(A, p, t) <= fairness(A, p, t-1) then
      increase = !increase
    end if
  else
    if fairness(A, F) < TMin then
      increase = false
    else if fairness(A, F) > TMax then
      increase = true
    else if fairness(A, F, t) <= fairness(A, F, t-1) then
      increase = !increase
    end if
  end if
  if increase then
    quota(A, p, t+1) = min(C, quota(A, p, t) + E)
  else
    quota(A, p, t+1) = max(0, quota(A, p, t) - E)
  end if
end for
```

The capacity control algorithm is executed for one peer A, and for each other peer p in the federation F. At a given moment t, A will first check if p already hear an interaction between them where A has already provided some resources to p, and then there will be a value of pairwise justice of members greater than 0. If this value

is negative, i.e. A still does not yield anything for p then the global fairness value should be used to evaluate the interaction.

Defining which of the values of justice to use, be it pairwise justice or global justice, we arrive at the capacity control step of peer A, where two thresholds are defined for the justice value Tmin and Tmax (these values have no fixed definition, and can be defined by experiment or as the user of FD-Nof wishes), where if the justice value is less than Tmin, the peer A will reduce the maximum amount of resources that it provides to the peer p, in order to increase justice in the future [1], if the justice value is above Tmax then A will increase the maximum amount of resources it can afford to p.

As a last case, if justice is between the maximum and minimum thresholds, then A will look at the past, comparing its present value of justice with the value of justice in the last iteration $t-1$, if the value of justice has evolved from the last iteration here [7], this means that the step following in the last iteration has had a positive effect (whether it is to increase the maximum amount of resources provided to the peer or decrease it) then it must be repeated, if it has resulted in a reduction of justice, then an inverse action of the previous step must be performed.

2.2 Fogbow - Architecture

The Fogbow federation building tool follows a distributed architecture coupled to the cloud provider.

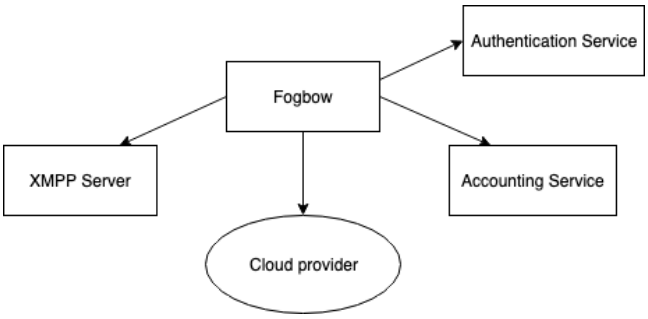

Figure 1: Fogbow Architecture

Fogbow functions as a manager, allocating and managing resources in the cloud, and interacting with other members of the federation so that each peer in a fogbow federation has its own Fogbow middleware deployment, thus adopting the P2P architectural model that is the same as FD-Nof.

In addition to the provisioning/management services, Fogbow implements a quota control mechanism and through the accounting component, which maintains a cloud usage report by the peers of the federation, thus allowing the implementation of a collection system either in monetary form or in the form of exchange of loans.

Fogbow also implements through XMPP (which is a secure communication protocol) client as a method of communication between Fogbow instances in connected components to different points of the federation, thus allowing the remote requests of resources between clouds.

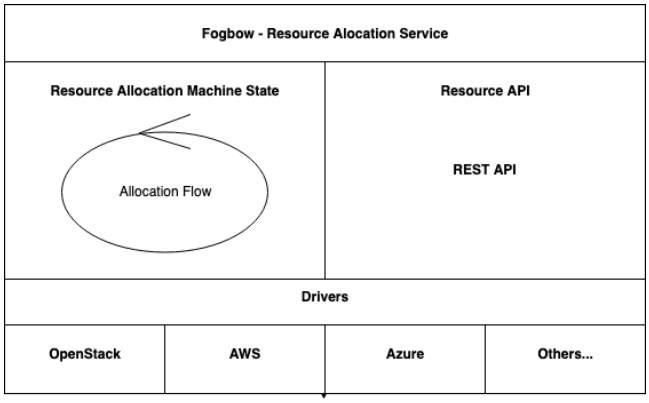

Figure 2: Fogbow Resource Allocation Service Architecture

The main component of the Fogbow is the resource allocation service, which is responsible for receiving the resource allocation requests and provisioning them in the Cloud to which it is connected using a state machine that defines step by step the provisioning process and its failure cases. The provisioning service, as well as all CRUD requests, is made available via a REST API, making it easy to access.

As its most interesting feature of this component is its adaptability to public and private cloud providers, defining a standard operating interface and implementing specific behaviors of each provider through drivers.

So everything, being a component with P2P architecture, having an easy-to-use interface and a secure communication model makes Fogbow a very interesting choice to be used in the standard FD-Nof implementation.

3 A BARTER-BASED MODEL WITH FOGBOW

This repository works as a mirror for application components:
https://github.com/GustavoDinizMonteiro/fd-nof

3.1 Purpose

The goal of this work is to create an implementation from the FD-Nof model bringing a solution to the fair resources exchange problem in a federation and free riders control.

Until then there was no implementation of the concept, making this first version a model for real test for FD-Nof and then study possible gaps and concept improvements. The proposal is also an adaptable, performance-oriented, resilient and designed for easy maintenance and evolution, requirements for a modern application.

The next sections of this work explain the architectural and design decisions that were made for implementation so that these requirements were fully met.

3.2 Architecture

To implement the FD-Nof using Fogbow middleware as the standard tool for provisioning a federation, and to maintain the same architectural and behavioral aspects with other possible federation

or cloud providers was defined a service interface provided by FD-Nof, therefore the integration with the federation/cloud provider through drivers that follow this interface.

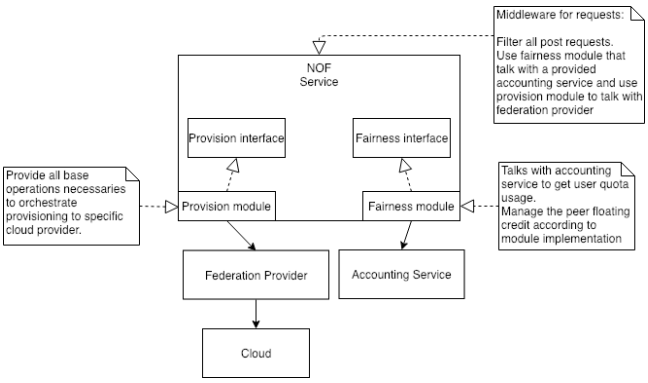

Figure 3: NOF Architecture

The architectural model of FD-Nof is based on the definition of two service interfaces, the first is the provisioning interface, which defines the steps of federation favors that will be better explained later. The second is the resource accounting interface that will be used by the capacity controller to control justice and the quota offered by that peer to the others.

These interfaces define that the drivers will have a communication role with the FD-Nof client so that they do not interfere in the capacity control and provisioning choices directly, thus avoiding deviations from the standardization of behavior among FD-Nof members.

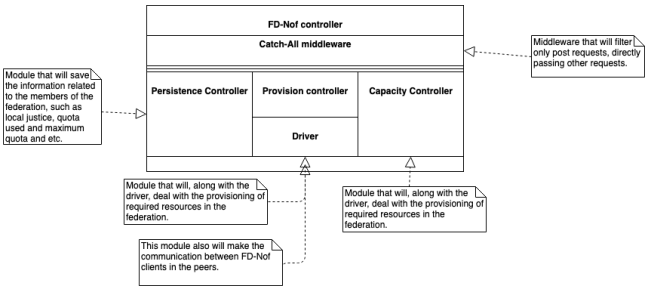

Figure 4: NOF Controller

The FD-Nof implementation has the requests filter at the top, which is responsible for evaluating all resource creation requests and passing them to the provisioning controller or the message handler, to transfer it to another point on the network if necessary. The provisioning controller will use the proprietary driver for resource creation, triggering the persistence controller to fetch necessary information, and then trigger the capacity controller that will take the necessary action for quota control and fairness.

The architectural model of FD-Nof also defines that it will be an independent component, being pluggable as a service, facilitating its adoption.

The driver implements the following interface that will allow FD-Nof to handle the requests.

- **getRequesterFromOrder**: extract order requester.
- **getProviderFromOrder**: extract the target member of the order.
- **createLocalResource**: Create resource from order in local member cloud.
- **dispatchRemoteRequest**: Resend an order to another member of the network.
- **getCurrentOrdersFromMember**: Get information on all resources currently allocated in the cloud for a particular other member.
- **preemptOrder**: Deallocate a specific feature of a remote member.
- **getCurrentQuotaUsedByMember**: Get resources amount currently being used by a remote member.

This interface that enables the implementation of drivers was the main architectural decision of the project, even though it was done with its default implementation with Fogbow, that name this work, the intention is that it can be used with any other technology, something that with the application code being made available in a public repository and with the adequate contribution license can have its use and its expansion made by anyone.

The Fogbow driver implements this interface in each method by making HTTP calls to pick up the information needed in Fogbow API directly and in the other two components as shown below, these components are implemented in this work and are now part of the Fogbow ecosystem, they are the message service and accounting service. The first one is used for sending messages between federation peers, thus enabling remote requests for resources, the second for monitoring the cloud's overall quota, the quota for use by the federation or a specific peer.

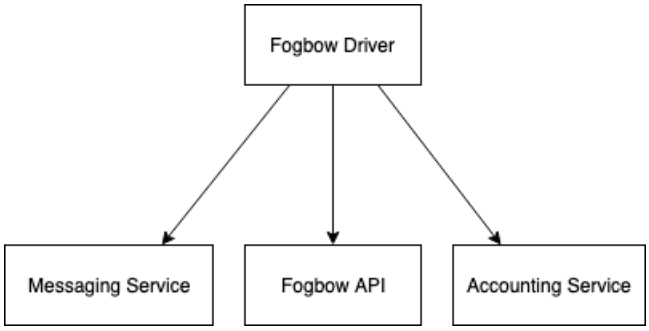

Figure 5: Fogbow Driver Architecture

This structure will be used by FD-Nof in the algorithms described in the next session for menage resources, quota calculation, and request routing.

3.3 Algorithms

The FD-Nof functions as a layer above the peer receiving all the requests and passing directly to the peer all non-resource creation requests. For re-creation requests, they can follow, according to their requester and destination, the following cases.

	Local peer	Remote peer
Local peer	case 1	case 2
Remote peer	case 3	X

3.3.1 Case 1. This is a request from the peer client itself for its cloud, so it does not require any FD-Nof processing as soon as it is directly passed on to the peer.

3.3.2 Case 2. This is a request in which the client of the local peer is requesting a resource in a remote member, soon this request will be passed by FD-Nof to the peer destination through the service of transfer of requests that the driver of the provider must implement. The transfer of this request would generate in the remote peer a remote request that is the case 3.

3.3.3 Case 3. When there is a request for a remote peer to the local peer, i.e. requesting a favor, FD-Nof will assume the role of handling that request.

For this purpose, FD-Nof use the information acquired through the quota service to calculate the maximum quota amount and provided for each other peer of federation, as well as their values of pairwise justice and in the case of the non existence so far from a pairwise justice value valid for interaction to another member (that is, the local peer is still not capable of anything to the other so that pairwise justice is negative) then global justice is used.

With this quota information provided and maximum quota in addition to the fair value, the exchange of resources in the federation is defined in two parts, the first of them defined as a peer receiving this favor request tries to allocate this resource in its cloud.

The FD-Nof works so that once a new request for resource allocation from another peer arrives, the FD-Nof will first check if peer has enough remaining quota within its total quota to meet the request, if yes to FD-Nof will try to create the resources in the local peer, in case of failure that results in resource cannot be allocated in any way will be returned a message describing the situation.

If the attempt to create the resource locally is successful, information about the resource will be returned to the requesting peer, but if for lack of resources available in the cloud was not able to create resource, it will check if there are other pairs that have a pairwise justice balance smaller than the peer owner of the new request, if there are other peers in such a situation then their resources will be preempted one by one, in order of the lowest value of justice for the greater and each preemption will be tried again create requested resource, this operation will be repeated until the request is answered or there are no more resources allocated from those peers with a lower justice value.

I even with the preemption of these resources, there is still insufficient quota for allocating this resource, the members with lesser value of justice will be listed again and will be sequentially passed on to each one the request as a remote request, that remote request will receive a special markup that says it's a forwarding.

In the second algorithm, it is demonstrated how the allocation of a remote request marked as forwarding is made. It is defined differently so that a request is not passed on more than once, to avoid a flood of the network, cases of indefinite forwarding times may also cause a large delay in obtaining a request response affecting the service quality, it was decided for this implementation, with better response instead infinite allocation attempts is more desirable.

Algorithm 2 Describe how to FD-Nof allocate a resource on local peer.

```
if requester has quota then
  success = createResource()
  if success then
    executeCapacityController()
    return RESOURCE
  else
    members = getMemberWithSmallerQuota()
    for member in members do
      for resource in member.resources do
        preempt(resource)
        success = createResource()
        if success then
          executeCapacityController()
          return RESOURCE
        end if
      end for
    end for
    members = getMembersWithHigherDebts()
    for member in members do
      success = createRemote()
      if success then
        executeCapacityController()
        return RESOURCE
      end if
    end for
    return ERROR
  end if
else
  return ERROR
end if
```

When the FD-NOF receives a remote request marked as forwarding, creating this resource also alters the fairness between the peer receiving the request and the remote peer, if it can handle the request. Upon receiving the request, it checks whether the first recipient of the request has a quota available within its maximum quota to meet that request, if yes, attempts to create the resource, if not, return to the original peer of request a message saying cannot attend to it.

If the creation attempt was made and failed because of insufficient quota in the cloud, the same process of preemption member resources with the value of justice less than justice between the peer who forwarded the request, in the same way as if even with the preemption of these resources it is still not possible to allocate resource of the request, it will return to the original peer a message saying that cannot be attended to.

These two algorithms are complementary and describe how the application works, using driver services interface to manage the resources reactively, that is, only when a request for resources leads to this.

Algorithm 3 Describe how to FD-Nof allocate a resource from a forwarded request on local peer.

```
if has quota then
  success = createResource()
  if success then
    executeCapacityController()
    return RESOURCE
  else
    members = getMemberWithSmallerQuota()
    for member in members do
      for resource in member.resources do
        preempt(resource)
        success = createResource()
        if success then
          executeCapacityController()
          return RESOURCE
        end if
      end for
    end for
    return ERROR
  end if
else
  return ERROR
end if
```

3.4 Technologies

The FD-Nof controller implementation was developed using Python language, which was chosen for its flexibility and development agility.

The application was built with Flask microframework, which has stability characteristics needed, un-opinionated, that is, bringing a minimum set of features needed and allows the developer and trusts the developer to make the right decisions and puts more control in their hands.

In the data layer the database used is PostgreSQL, a relational solution was chosen for all security that this brings the transactions and structuring of the data that are desired characteristics in the context of the application. PostgreSQL was chosen especially for being open source and very performance oriented, meeting high demands if necessary. To connect to the database was used SQLAlchemy tool, a SQL toolkit and Object Relational Mapper that allows structure tables and queries with the flexibility and good performance.

The Python language used to build FD-Nof, that although it is very flexible and high abstraction level, paid for it in its performance, however, the python community itself developed strategies to make the language more performative, one of them is the use Pypy of an alternative to the standard language interpreter CPython. Pypy is a Just In Time compiler that makes code execution much faster according to several benchmarks and is the recommended tool for execution in case the performance requirements are higher or there is a federation with several peers very large, with the consideration that this should add the Pypy to deploy the stack. CPython, however, is a built-in option present on UNIX systems and does not need additions to deploy stack with acceptable performance.

FD-Nof offers 3 deployments ways, the first through Docker container that makes the application work in the same way as the development environment whatever the deploy environment, this alternative has many advantages because of its system flexibility and environment configurations in which deploy and the current popularization of the containers will be done besides that the image created for application is more performative for using Pypy. The second through the Pipenv environment, which unites the application isolation of the Virtualenv python with the dependency management of Pip which facilitates the deployment of the application and can be done with a few commands, and the third in a native way in the machine, using common Python tools.

The Fogbow specific implementation of messaging and accounting services used by his driver was developed in Java with the SpringBoot framework, this choice was made to keep up with the stack of Fogbow project technologies that are completely made with Java technologies.

The SpringBoot Framework brings a convention load on configuration and platform popularity itself, which makes it a good choice of technology to maintain and expand the application, as well as being within the Java ecosystem that is quite complete. The Spring Boot was used to create the API rest that exposes the system functionalities implemented in a stateless service layer to allow greater scalability and uses an abstracted DAO by SpringBoot through JPA Repositories, this service layer brings the whole logic of the applications modularized.

In the data layer of this services PostgreSQL database was chosen, which besides the reasons mentioned above for choosing it in the FD-Nof controller it is also the one used in Fogbow, which makes it the ideal choice.

The Fogbow messaging component uses the XMPP protocol, which defines a secure private communication method provided by Jamppa client, an open source java tool that enables communication through this protocol.

4 CASE STUDY

The tool has been deployed and is now available for use, with its Fogbow driver it becomes a service most available to the organizations that use the platform, with the expectation that it will be used soon. But before that, at the end of the process of developing the scope of this work was together with the Fogbow team tested the following use cases.

4.0.1 Case 1. This is the optimal situation in FD-Nof, where one peer receives a resource request from another and there is a cloud quota to fulfill a request without the need to delete any other resource.

4.0.2 Case 2. FD-Nof receives a resource request from another peer and there is not enough cloud quota to meet the request, so is the need to preempt resources from other peers.

4.0.3 Case 3. FD-Nof receives a resource request from another peer and there is not enough quota in the cloud to fulfill the request even after preempting resources from other peers, thus having to pass on to other peers the request.

These other peers, in this case, will have enough quota to fulfill the request without preempting resources.

4.0.4 Case 4. FD-Nof receives a request and because of a lack of resources forward request, the peer receiving this transfer is also suffering from a lack of resources in the cloud and has to delete resources from other peers to fulfill this request.

The simulation of these cases was successful and made with two different cloud providers (OpenStack and Opennebula), both using Fogbow middleware.

This case study showed how useful the tool can be to expand the resources to which it has access, in cases such as high demands or in low demands where resources can be borrowed, it is effective. In this case study, the study of the functioning of justice control algorithms with the implementation was also planned, but this test was reconsidered since the formal proof of the correctness of this concept already existed.

5 EXPERIENCE AND LESSONS LEARNED

In this work i was able to learn a lot from the research work, something i had never done before that way, searching for scientific publications and references, testing comparing implementations of concepts, something different from the routine of implementing commercial applications with a predefined scope to which I was accustomed.

Something new for me and what I consider great learning was the study of new technologies such as the XMPP protocol and the use of new architectural models, such as the polyglot architecture used in the implementation of the Fogbow driver for FD-Nof, using different technologies into separate components and communicating through HTTP requests.

Other learning experiences that I can highlight in the planning and production of an article, a relatively long-term work that required planning, mainly due to unforeseen during the work and the learning about the planning of this type of work was valuable as well as the experience of writing and review the article.

6 DEVELOPMENT PROCESS

The FD-Nof development process can be described in 3 steps, in the first step the research was done on the concepts that involve NF-Nof, such as justice and provisioning algorithms and the P2P architectural model. At this stage, the work was focused on documentation of concepts to make it easier to use them later in both the implementation and the description of them in the article.

In the second stage, the architectural model of the application was elaborated, choosing technologies to be used, planning the integration between the components and how to meet the non-functional requirements of the application, such as performance.

In the third stage was where the development of the application code happened, together with the Fogbow team who helped a lot in the process of building the Driver, working on a gradual cycle of implementation and testing of the application paused sometimes for replanning in the face of some difficulties or changes in the scope of the solution..

7 KEY CHALLENGES AND THEIR SOLUTIONS

During the development process, there were significant delays in the development of the driver in this step compared to the planned, due to the communication model initially planned between the peers of the Federation had to be replaced by a more secure one, which was the XMPP protocol. This added a new difficulty, because the known client meeting the requirements for XMPP was developed only for Java platform and FD-Nof is built in Python, the solution to this was to reshape the driver by extracting part of its functionality into separate components that were developed in Java, replacing those implementations in the driver for HTTP calls to those services that provide the functionality. Thus the planning of the development stage had to be redone.

Another point that could be put as a difficulty for the job was the infrastructure required to test the application, a Fogbow deploy with a cloud with resources available for management, which was solved by the Fogbow team and the Laboratory support team distributed systems that provided this structure.

8 RELATED WORK

From the conceptual model perspective of the network of favors, many approaches have already been explored to promote collaboration in peer-to-peer systems. One is based on the reputation that by recording information about the past behavior of other peers, one can make decisions about collaboration. In P2PRep [9] and EigenRep [8], each pair locally stores information about its interactions with other pairs and also collects information from other pairs, which are then used to determine your reputation or credit. Such strategies are sufficient to promote collaboration and discriminate against free riders.

Satsiou and Tassiulas [13] also proposed the reputation-based scheme that dynamically controls the availability of file-sharing systems in which the bandwidth of the peers is shared between the download and upload resources. This ensures the fairness of fellow collaborators by ensuring that they receive resources in proportion to their contributions.

To federated clouds, similar to our strategy to maximize benefits, Goiri et al. [3] have attempted to achieve the same goal of estimation if it is better to outsource, insourcing or simply deactivate idle resources. Mihailescu [10] and Teo and Gomes et al. [4], Adopt the strategy of dynamic prices to improve the benefits of the federation. In both works, there is in freeriding since the approaches are market-based. As their main objective is to maximize benefits, they share ideas similar to those of FD-Nof, but as the FD-Nof mechanism is completely non-monetary, costs are reduced, benefits increase and free riders are isolated by regulation resources.

The RESERVOIR [12] project uses policy-oriented strategies to ideally find the best positioning of virtual machines for physical machines, considering economic, performance and availability aspects. This work does not address the problem of free riding since past interactions do not leverage future negotiation decisions. In addition, while our approach is P2P, and therefore two private clouds need to unite autonomously to the federation to be able to exchange resources, in the RESERVOIR the collaboration between two clouds is established in advance by a structured agreement that specifies the capacity in terms of number and size of virtual machines available along with other restrictions, such as cost, quality of service, and security level. These characteristics bring limitations to the federation, a significant disadvantage, because the larger the

federation, the greater the chances that the peak and peak patterns of different members will match.

In order to define a reciprocity model and to encourage the exchange of resources, we can cite works such as that of E. FalcÃčo [2], where a technique of reciprocity is defined that inspired resource allocation algorithms 2 3 where we have the transfer requests among peers as way to improve service quality and to increase justice among all the peers involved, because in this work there is still no known implementation in use, making this work an adaptation of it together with the FD-Nof concept.

In view of these points and given that the concept of FD-Nof was already defined, this work differentiates itself by bringing an implementation, defining an architectural model that is easy to deploy and adding the minimum elements to the customer's stack and new points in the resource allocation algorithms that increase the interaction between peers in the network and which can lead to better values of fairness among peers.

9 CONCLUSION

In this work, an implementation was developed for the FD-Nof concept previously defined, with addition of algorithms for reciprocal provisioning of resources by the network, enabling the implementation of a cloud federation with a non-monetary financial model, which simplifies the application and requisition of resources between the federation pairs makes it feasible for small and medium organizations that could not afford monetary costs.

With this tool, which focuses on its ability to support any cloud or federation provider, bringing beyond its default implementation with Fogbow middleware the possibility of client implements proprietary drivers for other federation middleware providers as directly cloud providers, it becomes possible to implement FD-Nof and leverage resource switching model to maximize cloud efficiency and service quality in high demand and maintain and regulate the exchange resources in a fair way to stop free riders actions.

Through the chosen technologies, the tool also ensures a good performance with requests service time limited by infrastructure aspects, such as network and database, with FD-Nof itself having a small impact on performance as suggested, a large poll of requests can be solved with the horizontal scale application and dividing the requests between the nodes of the cluster with a load balancer, for example, in an easy way, since the FD-Nof is a stateless application, making possible application could attend large dimensions of requests even though they cause request forwarding, which are the cases with the longest response delay due to the routing process.

10 LIMITATIONS AND FUTURE ENHANCEMENTS

The main pending point in FD-Nof is the provision of a public deploy of the publication, given that for a real situation it needs a Fogbow deployment and resources to be managed. Also are missing implementation of a management tool where it is possible to manage registers, visualize statistics of interaction in the network, quota values, and justice of other peers.

Another pending point is to create richer documentation of tools, to understand all the points described in this work.

Other limitations are the non-support for Python 2.7, which although it is about to become deprecated is still widely used and comes as standard on various Linux distributions, testing, and possible fixes to support other database systems as well as testing the adaptations necessary for full operation on a Windows system.

At future study points, we can add security tests that were not in the scope of this work and a detailed analysis of how FD-Nof performance with many large networks, and also with the possibility of a greater number or an infinite number of forwarding requests.

11 ACKNOWLEDGEMENTS

I thank my family first for supporting me at all times and for all my fellow graduates who have often helped and supported me, especially my mother, for all the effort she devoted to my education.

Thanks also to teachers Matheus Gaudencio and Thiago E. Pereira for believing in my work and taking me to make my first experience in a development project, a high-level work, in which I learned immensely.

I also have a lot to thank everyone I worked with on development projects especially Walter A. Alves, Saulo S. Toledo, and Ricardo A. Santos, for all the knowledge they gave and the patience to do so.

And finally, I thank the Fogbow team and the Support team at the Distributed Systems Laboratory, which provided me the technical and infrastructure support needed to complete this work.

REFERENCES

[1] Cirne W Mowbray M. Andrade N, Brasileiro F. 2004. Discouraging free riding in a peer-to-peer cpu-sharing grid. In: High performance distributed computing. *IEEE international symposium* 13, 13 (2004), 129 – 37.

[2] Jose Luis Vivas. Eduardo de Lucena Falcao. Francisco Brasileiro, Andrey Brito. 2016. Enhancing P2P Cooperation through Transitive Indirect Reciprocity. *International Conference on Distributed Computing Systems Workshops* 16 (2016), 189 – 198.

[3] Torres J Goiri I, Guitart J. 2012. Economic model of a cloud provider operating in a federated cloud. *Inf Syst Front* 14, 4 (2012), 827 – 43.

[4] Kowalczyk R. Gomes ER, Vo QB. 2012. Pure exchange markets for resource sharing in federated clouds. *Concurrency Compute* 24, 9 (2012), 91–977.

[5] Buyya R. Grozev N. 2014. Inter-cloud architectures and application brokering: taxonomy and survey: taxonomy and survey. *Software* 44, 3 (2014), 369–90.

[6] Parekh S Tilbury DM. Hellerstein JL, Diao Y. 2004. Feedback control of computing systems. *New Jersey: John Wiley Sons* (2004).

[7] Staddon JER. Hinson JM. 1983. Matching, maximizing, and hill-climbing. *J Exp Anal Behav* 40, 3 (1983), 321 – 31.

[8] Garcia-Molina H Kamvar SD, Schlosser MT. 2002. The eigentrust algorithm for reputation management in p2p networks. *Working Paper - Stanford InfoLab* (2002), 2002 – 56.

[9] Managing and sharing servents' reputations in p2p systems. 2003. Inter-cloud architectures and application brokering: taxonomy and survey. *IEEE Trans Data Knowl Eng* 15, 4 (2003), 840 – 54.

[10] Teo YM Mihailescu M. 2010. Dynamic resource pricing on federated clouds. In: Cluster, cloud and grid computing (CCGrid). *IEEE/ACM international* 10 (2010), 513 – 17.

[11] W. Cirne N. Andrade, F. Brasileiro and M. Mowbray. 2007. Automatic grid assembly by promoting collaboration in peer-to-peer grids. *J. Parallel and Distrib. Comput.* 67, 8 (2007), 957–966.

[12] Epstein A Hadas D Loy I Nagin K et al Rochwerger B, Breitgand D. 2011. Reservoir - when one cloud is not enough. *Computer* 44, 3 (2011), 44 – 51.

[13] Tassiulas L Satsiou A. 2010. Reputation-based resource allocation in p2p systems of rational users. *IEEE Trans* 21, 4 (2010), 466 – 79.