Cloud Computing System Based on a DHT Structured Using an Hyperbolic Tree

Telesphore Tiendrebeogo

Polytechnic University of Bobo-Dioulasso
Email: tetiendreb@gmail.com

Oumarou Sié

University of Ouagadougou
Email: oumarou.sie@gmail.com

***Abstract*—During the last decade, Cloud Computing (CC) has been quickly adopted worldwide, and several solutions have emerged. Cloud computing is used to provide storage service, computing power and flexibility to end-users, in order to access data from anywhere at any time. Thus, Cloud Computing is a subscription-based service where you can obtain networked storage space and computer resources. The cloud makes it possible for you to access your information from anywhere at any time. Distributed Hash Table (DHT) plays an important role in distributed systems and applications, particularly in environments distributed on a large scale. In the model of normal Client/Server (C/S model), as we centralize most of the resources on the server, it becomes the most important part as well as the bottleneck and the weak point of the system. On the contrary, distributed model (a typical is Peer-to-Peer (P2P) model) distributes the resources on the nodes in the system. In this paper, we propose a new system of Cloud Computing based on our DHT structured using an hyperbolic tree, which organizes the distributed services so well that peers only need to know part of the system they can get services efficiently. DHT provides two basic operations: retrieves service from DHT and stores service into DHT, which is so simple and graceful, but is suitable for a great variety of resources (Applications, Infrastructures, Platforms), and provides good robustness and high efficiency, especially in large-scale systems. Resources as services are distributed by using virtual coordinates taken in the hyperbolic plane. We use the Poincaré disk model and we perform and evaluate our cloud structure performances. First, we show that our solution is scalable, consistent, reliable. Next, we compare the performances realized by the substitution strategy, which we propose with the strategy of classic replication in a dynamic context.**

Keywords–Cloud Computing; DHT; Hyperbolic Tree; Consistent; Reliable; Cloud Services; Storage; Discovery.

I. INTRODUCTION

For decades, extensive work has been done for Distributed Hash Table (DHT) [1] [2] [3] [4]. In academia, researchers have proposed several variants of DHT and improvements, which manage the resources in many kinds of structures, providing abundant choices for the construction of distributed system. Cloud computing (CC) has recently emerged as a compelling paradigm for managing and delivering services over the internet. The rise of Cloud computing is rapidly changing the landscape of information technology, and ultimately turning the long-held promise of utility computing into a reality. The latest emergence of Cloud computing is a significant step towards realizing this utility computing model since it is heavily driven by industry vendors. Thus, Cloud computing centre provides on-line storage and retrieving functionalities, and the services are distributed to tens of thousands of machines in a distributed way.

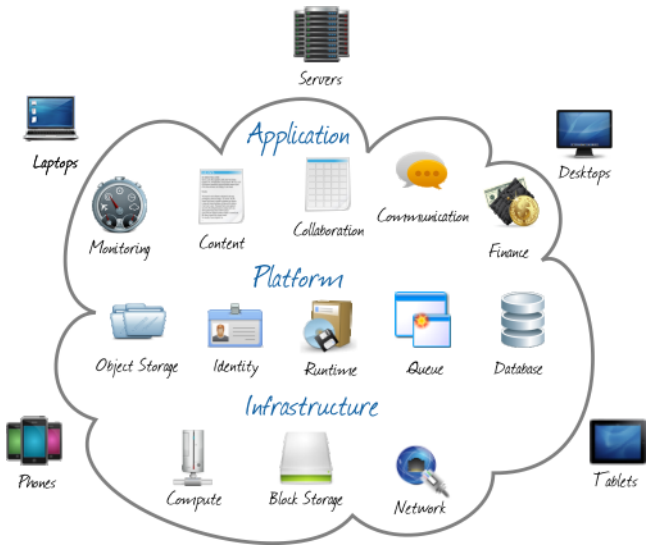

Figure 1. Cloud computing model [5].

In Figure 1, it is important to notice that many devices can interact between them by services exchanges. These services can be as well diverse as varied, going of the monitoring to the network via databases.

Therefore, a distributed storage system seems to be a good choice to manage the gigantic storage. Many big players in the software industry, such as Microsoft, as well as other Internet technology heavyweights, including Google and Amazon, are joining the development of cloud services [6] [7] [8] [9] [10] [11]. Several businesses, also those not technically oriented, want to explore the possibilities and benefits of CC [12]. However, there is a lack of standardization of Cloud computing and Services (CCSs) [7] [8] [13], which makes interoperability when working with multiple services or migrating to new services difficult. Further, there is a big marketing hype around CC, where on-line service providers re-brand their products to be part of the cloud movement [14].

There are a certain number of characteristics associated to the Cloud computing, as follow: variety of resources, Internet centric, virtualization, scalability, automatic adaptation, resource optimization, service SLAs (Service-Level Agreements) and infrastructure SLAs [15]. The Cloud computing corresponds to a virtual computation resource with the possibility to maintain and to manage by itself. From a structural point of view, the resources may be (Figure 2):

1) IaaS (Infrastructure as a Service).
2) PaaS (Platform as a Service).
3) SaaS (Software as a Service).

In Figure 2, the red layers represent layers managed by the providers of cloud resources and the green layers represent those managed by the users of services. Thus, it presents the services characteristics as shown by Chunye et al. [16]. Some defining characteristics of SaaS include:

1) Web access to commercial software.
2) Software is managed from a central location.
3) Software delivered in a "one to many" model.
4) Users not required to handle software upgrades and patches.
5) Application Programming Interfaces (APIs) allow for integration between different pieces of software.

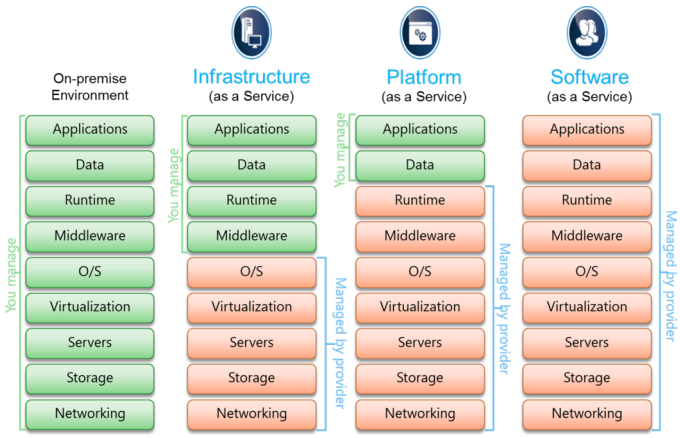

Figure 2. Cloud services model [17].

There are a number of different takes on what constitutes PaaS, but some basic characteristics include:

1) Services to develop, test, deploy, host and maintain applications in the same integrated development environment. All the varying services needed to fulfill the application development process.
2) Web based user interface creation tools help to create, modify, test and deploy different User Interface (UI) scenarios.
3) Multi-tenant architecture where multiple concurrent users utilize the same development application.
4) Built in scalability of deployed software, including load balancing and failover.
5) Integration with web services and databases via common standards.
6) Support for development team collaboration - some PaaS solutions include project planning and communication tools.
7) Tools to handle billing and subscription management.

As with the two previous (SaaS and PaaS), IaaS is a rapidly developing field. That said, there are some core characteristics, which describe what IaaS is. IaaS is generally accepted to comply with the following:

1) Resources are distributed as a service.
2) Allows for dynamic scaling.
3) Has a variable cost, utility pricing model.
4) Generally includes multiple users on a single piece of hardware.

Thus, for example, it may realize them for a lot of large-scale server cluster structures, including computation servers, storage servers, the bandwidth resources [18]. Cloud platform permits to manage both a large number of computer resources and to store a large number of data. Resource allocation is made in Cloud platform such as user feel that he uses an infinitive amount of resources. In this paper, we make the following contributions:

1) We introduce a new structure of Cloud computing using an hyperbolic tree, in which each node is associated with a resource server's and takes of virtual coordinates into hyperbolic space of the model of the Poincaré disk.
2) We show how Cloud infrastructures can communicate by greedy routing algorithm using [19].
3) We present naming and binding principle in our solution.
4) We perform some simulations, furthermore:
   - We show that this structure provides scalability and consistence in database services like data storage and retrieving.
   - We present the results concerning the Floating point precision.
   - We propose a new strategy of resistance in the phenomenon of churn, which we called substitution strategy and we show that it allows to reach satisfactory results in terms of the success rate of the storage and lookup queries of services. Indeed, this strategy allows to improve the availability of the resources on the servers of services.

The remainder of this paper is organized as follows. Section II gives a brief overview of the related previous work. Section III highlights some properties of the hyperbolic plane when represented by the Poincaré disk model. Section IV defines the local addressing and greedy routing algorithms of cloud computing system. Section V defines the binding algorithm of our Cloud system. Section VI presents the results of our practicability evaluation obtained by simulations and we conclude in Section VII.

II. Related work

The main differences between the CCSs that are deployed are related to the type of service offered, such as storage space and computing power, platforms for own software deployment, or online software applications, ranging from web-email to business analysis tools. Cloud infrastructure services typically offer virtualization platforms, which are an evolution of the virtual private server offerings that are already known for years [20]. The offers of Cloud platform services include the use of the underlying infrastructure, such as servers, network, storage or operating systems, over which the customers have no control, as it is abstracted away below the platform [20] [21]. Cloud software offerings typically provide specific, already-created applications running on a cloud infrastructure, such as simple storage service [22]. A very well known SaaS is the web-based e-mail. Most software CCSs are web-based

applications, which can be accessed from various client devices through a thin client interface, such as a web browser. An Internet-based storage system with strong persistence, high availability, scalability and security is required. Obviously, the centralized methods is not a good way because it lacks of scalability and has the single point of failure problem. If the centre fails, all the owners lose the capability to access their data, which may cause inestimable losses. Besides, it is impossible to store all the data on one machine, though it is facility for management. Even in the cloud computing centre, which provides on-line storage functionality, the data is distributed to tens of thousands of machines in a distributed way. Therefore, a distributed storage system seems to be a good choice to manage the gigantic storage.

How to organize so many kinds of data efficiently is the first hit. DHT with its wonderful structure is suitable to the distributed environment. DHT provides a high efficient data management scheme that each node in the system is responsible to a part of the data. It supports exact and quick routing algorithm to ensure users retrieving their data accurately and timely. Furthermore, replication and backup, fault-tolerant and data recovery, persistent access and update, which are concerned in the storage area are not difficult to DHT. Recently, many researches have proposed a lot of systems such as Chord based Session Management Framework for Software as a Service Cloud (CSMC) [23], MingCloud based on Kademlia algorithm [24], Search on the Clou which is built on Pastry [25]. These systems are based on the DHT structure such as MSPastry [26], Tapestry [27], Kademlia [28], CAN, and Chord [29].

Furthermore, an Efficient Multi-dimensional Index with Node Cube for Cloud computing system [30] has been proposed by Xiangyu Zhang et al., also the RT-CAN index in their Cloud database management system in 2010 [31] has been built by Jinbao Wang et al. Both these two schemes are based respectively on k-d tree and R-tree.

Our work follows the framework proposed in [32]. However, to support multi-dimensional data, new routing algorithms and storage and retrieve queries processing algorithms are proposed. Furthermore, our indexing structure reduces the amount of hops for transfer inside the Cloud and facilitates the deployment of database back-end applications.

III. Practical use of the hyperbolic geometry

In this section, we present some properties concerning hyperbolic geometry. Geometries Hyperbolic show similarity with regard to the Euclidian geometry. Both have the same concepts of distances and angles, and they have many common theorems. The two-dimensional hyperbolic plane is the simplest hyperbolic space and has a constant negative curvature equal to -1 as opposed to the Euclidean space, which is not curved. The model that we use to represent the hyperbolic plane is called the Poincaré disk model. In this model, each point is referred by complex coordinates. Beardon, Kleinberg, and Krioukov have detailed all the concepts necessary to understand the hyperbolic plane [33]–[35].

Next, each line of $\mathbb{H}^2$ splits the plane in several areas as in the Euclidean plane, but there are a certain number of differences. In the Euclidean space, one of elementary properties is the impossibility to create more than two half planes without having them intersect. Our embedding is based on the geometric property of the hyperbolic plane, which allows to create distinct areas called half planes. As explained by Miquel in [36], in the hyperbolic plane, we can create n half spaces pair wise disjoint whatever n. Our embedded algorithm is based on this property (red line in Figure 3).

Another essential property is that we can split the hyperbolic plane with polygons of any sizes, called p-gons. Each tessellation is represented by a notation of the form $\{p, q\}$ where each polygon has p sides with q of them at each vertex. This form is called a *schläfli symbol*. There exists a hyperbolic tessellation $\{p, q\}$ for every couple $\{p, q\}$ that satisfy the following relation: $(p-2)*(q-2) > 4$.

In a splitting, p is the number of sides of the polygons of the *primal* (the black edges and green vertices in Figure 3) and q is the number of sides of the polygons of the *dual* (the red triangles in Figure 3).

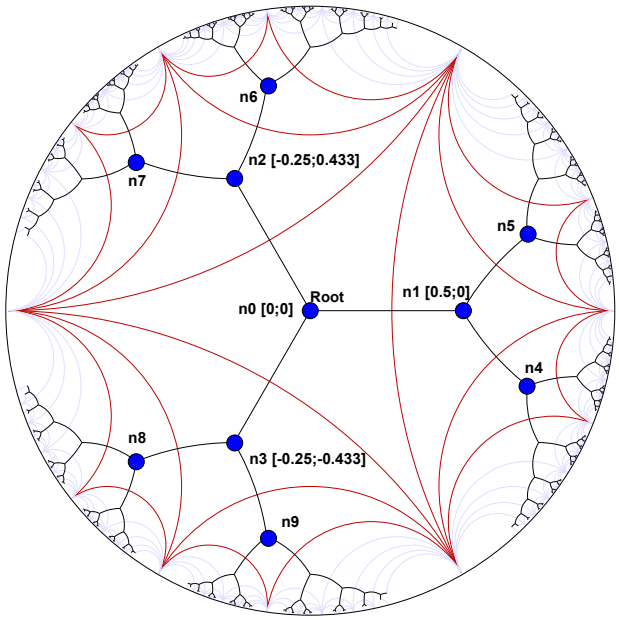

Figure 3. 3-regular tree in the hyperbolic plane.

Our method is to partition the plane and address each node uniquely. We set p to infinity, thus transforming the primal into a regular tree of degree q. The dual is then tessellated with an infinite number of q-gons. This particular tiling splits the hyperbolic plane in distinct spaces and constructs our embedded tree. Figure 3 is an example of a hyperbolic tree with $q = 3$ associated with the Poincaré disk model, in which the hyperbolic plane is represented by an open unit disk of radius 1. Thus, in this model:

- Points are represented by Euclidean points within this unit disk.
- Lines are arcs of circles intersecting the disk and meeting its boundaries at right angles.
- Distances between any two points z and w in the hyperbolic plane ($d_{\mathbb{H}}$) are given by curves minimizing the distance between these two points and are called geodesics of the hyperbolic plane.

To compute hyperbolic distance means to use the Poincaré metric, which is an isometric invariant:

$$d_{\mathbb{H}}(z,w) = argcosh(1+2\Delta) \quad (1)$$

with:

$$\Delta = \frac{|z-w|^2}{(1-|z|^2)(1-|w|^2)} \quad (2)$$

because:

$$sinh^2\frac{1}{2}d_{\mathbb{H}}(z,w) = \Delta$$

and

$$cosh^2\frac{1}{2}d_{\mathbb{H}}(z,w) = \frac{|1-z\bar{w}|^2}{(1-|z|^2)(1-|w|^2)}$$

For more details on the Poincaré metric we refer the reader to the proof in [33]. The hyperbolic distance $d_{\mathbb{H}}(z,w)$ is additive along geodesics and is a Riemann's metric. The authors of [33] do the sketch of an important property for greedy routing: the strict inequality in the triangle inequality. The following relation permits to compute this metric:

$$tanh\frac{1}{2}d_{\mathbb{H}}(z,w) = \frac{|z-w|}{|1-z\bar{w}|} \quad (3)$$

where $\bar{w}$ is the complex conjugate of w.

In theoretical perspective, the hyperbolic plane is unlimited. However, to carry out measurements, it is necessary to use a modelled representation of this plane and to define a precision threshold for the calculations.

IV. Greedy routing in the hyperbolic plane

In this section, we present how we create the hyperbolic addressing tree for cloud services storage and how we localize these latters in our cloud. We propose here a dynamic, scalable, and reliable hyperbolic greedy routing algorithm [37]. The first step in the creation of our cloud computing based on hyperbolic-tree of services nodes is to start the first services nodes and to define the degree of the addressing tree.

We recall that the hyperbolic coordinates (i.e., a complex number) of a services' servers of the addressing tree are used as the corresponding address to the services' servers in the cloud. A node of services of the tree can provide the addresses corresponding to its children in the hyperbolic-tree. The degree of this latter determines how many addresses each services' servers will be able to attribute for news servers of services connexions. The degree of the hyperbolic-tree is chosen at the beginning for all the lifetime of the cloud. Our cloud architecture is then built incrementally, with each new services' server joining one or more existing servers of services. Over time, the servers of services will leave the overlay until there are no more services' servers in the cloud. So, for every service that must be stored in the system, a Service's IDentifier (SID) is associated with it and map then in key-value pair. The key will allow to determine in which servers of services will be stored (like in the Section V). Furthermore, when a service is deleted, the system must be able to update this operation in all the system by forwarding query through the latter. This method is scalable and reliable because unlike [38], we do not have to make a two-pass algorithm over the whole cloud to find its highest degree. Also, in our system, a server can connect to any other server at any time in order to obtain an address.

The first step is thus to define the degree of the tree because it allows building the *dual*, namely the regular $q-gon$. We nail the root of the tree at the origin of the *primal* and we begin the tiling at the origin of the disk in function of q. Each splitting of the space in order to create disjoint subspaces is ensured once the half spaces are tangent; hence, the *primal* is an infinite q-regular tree. We use the theoretical infinite q-regular tree to construct the greedy embedding of our q-regular tree. So, the regular degree of the tree is the number of sides of the polygon used to build the *dual* (see Figure 3). In other words, the space is allocated for q child servers of services. Each services' server repeats the computation for its own half space. In half space, the space is again allocated for $q-1$ children. Each child can distribute its addresses in its half space. Algorithm 1 shows how to compute the addresses that can be given to the children of a services' server. The first services' server takes the hyperbolic address (0;0) and is the root of the tree. The root can assign q addresses.

Algorithm 1 Computing the coordinates of a services' servers children.

```
1:  procedure CalcChildrenCoords(server, q)
2:      step ← argcosh(1/sin(π/q))
3:      angle ← 2π/q
4:      childCoords ← server.Coords
5:      for i ← 1, q do
6:          ChildCoords.rotationLeft(angle)
7:          ChildCoords.translation(step)
8:          ChildCoords.rotationRight(π)
9:          if ChildCoords ≠ server.ParentCoords then
10:             StoreChildCoords(ChildCoords)
11:         end if
12:     end for
13: end procedure
```

Algorithm 2 Query's greedy routing algorithm in the cloud.

```
1:  function getNextHop(services_S, query) return services_S
2:      w = query.destServCoords
3:      m = services_S.Coords
4:      d_min = argcosh(1 + 2|m−w|^2 / ((1−|m|^2)(1−|w|^2)))
5:      p_min = services_S
6:      for all server_N ∈ services_S.Neighbors do
7:          n = server_N.Coords
8:          d = argcosh(1 + 2|n−w|^2 / ((1−|n|^2)(1−|w|^2)))
9:          if d < d_min then
10:             d_min = d
11:             p_min = server_N
12:         end if
13:     end for
14:     return p_min
15: end function
```

This distributed algorithm ensures that each services' server is contained in distinct spaces and has unique coordinates.

All the steps of the presented algorithm are suitable for distributed and asynchronous computation. This algorithm

allows the assignment of addresses as coordinates in dynamic topologies. As the global knowledge of the cloud is not necessary, a new services' server can obtain coordinates simply by asking an existing services' server to be its parent and to give it an address for itself. If the asked services' server has already given all its addresses, the new server must ask an address to another existing database server. Besides, when a new services' server obtains an address, it computes the addresses (i.e., hyperbolic coordinates) of its future children. The addressing hyperbolic-tree is thus incrementally built at the same time than the cloud.

When, a new services' server has connected to servers already inside the cloud and has obtained an address from one of those servers, it can start sending requests to store or lookup service in the cloud. The routing process is done on each services' server on the path (starting from the sender) by using the greedy Algorithm 2 based on the hyperbolic distances between the servers. When, a query is received by a services' server, this latter computes the distance from each of its neighbours to the destination and forwards the query to its neighbour, which is the closest to the destination (destination services' servers computing is given in Section V). If no neighbour is closer than the server itself, the query has reached a local minima and is dropped.

In a real network environment, link and services' servers failures can occur often. If the addressing hyperbolic-tree is broken by the failure of a services' server or link, we flush the addresses attributed to the servers beyond the failed server or link and reassign new addresses to those servers (some servers may have first to reconnect to other servers in order to restore connectivity). But, in this paper, we have not detailed this solution.

V. Naming and binding in the hyperbolic plane

In this section, we explain how our cloud system stores and retrieves the resources by using these latters names, which is mapped to its address (virtual coordinates where it is possible to find servers of services). Our solution uses a structured DHT system associated to the virtual addressing mechanism of servers of services and to the greedy routing algorithms presented in Section IV. At startup, each new resource server uses a name that identifies the service (Application, Platform, Infrastructure) that is shared in the system. This name will be kept by the resource server containing the service during all the lifetime of the cloud. When the new resource server obtains an address, it stores the names of these services on different others resources servers. This storage uses the structured DHT of our cloud to store a fragment of key obtain by hashing of service name (explain in the follow). If the same sub-key is already stored in the cloud, an error message is sent back to the resource server containing concerned service in order to change this service name. Thus, the DHT structure used in our cloud itself ensures that services names are unique.

A (name, address) pair, with the name mapping as a key is called a *binding*. Figure 4 shows how and where a given binding is stored in the cloud.

A binder is any resource server that stores these pairs. The depth of a peer in the hyperbolic addressing tree can be defined as the number of parent peers to go through for reaching the root of the tree (including the root itself). When the cloud system is created, a maximum depth for the potential binders is chosen. This depth permits to define the maximum number of servers of services that can connect and share different services. Also, the depth d is chosen such it minimizes the

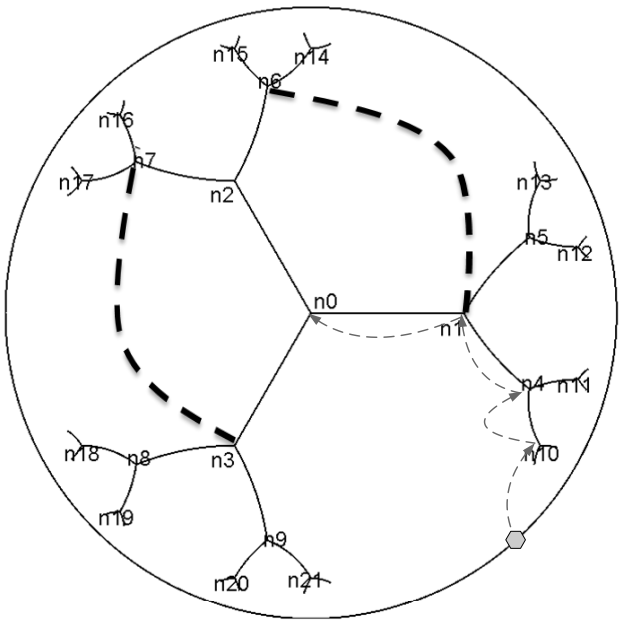

Figure 4. Hyperbolic cloud system.

Algorithm 3 Lookup algorithm in general context for inserting, deleting and updating of service

```
1: function LookupProcess(PrimeResourceServer, Query) return Service
2:     SID ← Tg.GetQuerySID()
3:     Key ← Hash(QuerySID)
4:     for A dol r ∈ R_Circulare
5:         d ← P_Max
6:         i ← 1
7:         while i ≤ ⌊1/2 × log(N)/log(q)⌋ && d ≥ 0 do
8:             TgServAd[r][d] ← GetValue(Key)
9:             Service ← GetValue(TgServAd[r][d], SID)
10:            if Service != null then
11:                if Query == delete then
12:                    delete(SID)
13:                end if
14:                if (Query == update) then
15:                    update(SID)
16:                end if
17:                if Query == select then
18:                    return Service
19:                    break
20:                end if
21:                i++
22:            end if
23:            d−−
24:        end while
25:    end for
26:    return 0
27: end function
```

Inequality (4):

$$p \times (\frac{(1-(p-1)^d)}{2-p} + 1 \geq N \quad (4)$$

with p the degree of our hyperbolic tree. Thus, this value is defined as the *binding tree depth*.

When a new resource server joins the cloud by connecting to other resources servers, it obtains an address from one of these resources servers. So, each service name of the resource server is mapped into key by hashing its name with the SHA-2 algorithm (SHA-2 gives 512-bit key). Next, the new resource server divides the 512-bit key into 16 equally sized 32-bit sub-keys (for redundancy storage). The resource server selects the first sub-key and maps it to an angle by a linear transformation. The angle is given by:

$$\alpha = 2\pi \times \frac{\texttt{32-bit sub-key}}{\texttt{0xFFFFFFFF}} \quad (5)$$

The resource server then computes a virtual point v on the unit circle by using this angle:

$$v(x,y) \text{ with } \begin{cases} x = cos(\alpha) \\ y = sin(\alpha) \end{cases} \quad (6)$$

Next, the resource server determines the coordinates of the closest binder to the computed virtual point above by using the given *binding tree depth*. In Figure 4, we set the *binding tree depth* to three to avoid cluttering the figure. It is important to note that this closest service (name of the binder) may not really exist. If no resource server is currently owning this address, this latter, then sends a stored query (containing service name and address) to this closest resource server. This query is routed inside the cloud by using the greedy algorithm of Section IV. If the query fails because the binder does not exist or because of node/link failures, it is redirected to the next closest binder, which is the father of the computed binder. This process continues until the query reaches to the root resource server having the address (0;0) (which is the farthest binder) or the number of resources servers is given by the following relation (radial strategy):

$$S \leq \left\lfloor \frac{1}{2} \times \frac{log(N)}{log(q)} \right\rfloor \quad (7)$$

with N equal to number of servers of services, q to degree of hyperbolic-tree.

This strategy permits to not saturate the closest servers of root's server of the hyperbolic tree. Indeed, when the replication of service go down towards the hyperbolic tree root's, the different paths from edge of unit circle towards root tighten and provoke a saturation of the servers of services. Inequality (7) permits to reduce this risk, and gives a better load balancing of services storage in our cloud. Thus, we search with this method, to increase the number of copies across a path of the hyperbolic tree. So, this strategy has a double interest:

- Allows to make the services available.
- Limits the saturation of the servers as a result of large number of services query and the same reduces the time of response to the servers.

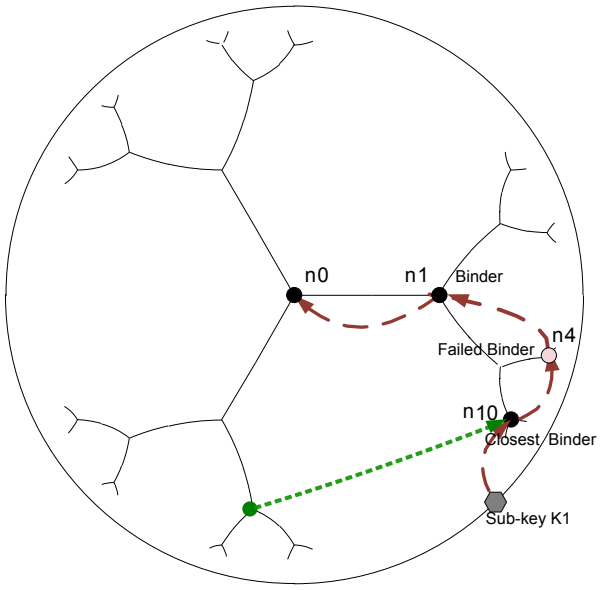

Figure 5. Radial replication in the cloud.

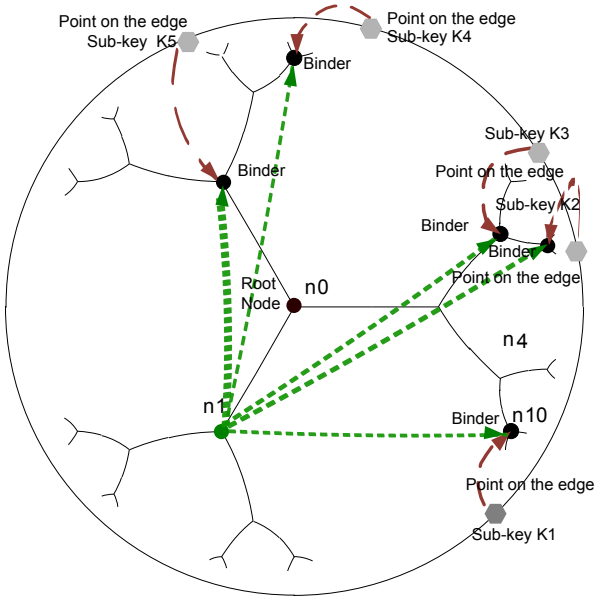

Figure 6. Circular replication in the cloud.

To reduce the impact of the dynamic (the leave or the join of the services server's also the adding or the deleting of the service) of the cloud, uses a redundancy mechanism that consists to increase the number of copies. The number of stored copies of a pair along the binding radius may be an arbitrary value (chosen to the cloud creating) set at the overlay creation. Similarly, the division of the key in 16 sub-keys is arbitrary and could be increased or reduced depending on the redundancy (circular strategy) needed.

To conclude, we can define two redundancy mechanisms (represented by Figure 5 and Figure 6) for storing copies of a given binding:

1) radial strategy of replication.
2) circular strategy of replication.

Algorithm 4 Services storage's processing in the general context

```
1: function StoreProcess(PrimeServServer Query) return 0
2:     SID ← Query.GetSID()
3:     Key ← Hash(SID)
4:     for A doll red ∈ R_Circular
5:         depth ← P_Max
6:         i ← 1
7:         while i ≤ ⌊1/2 × log(N)/log(q)⌋ && depth ≥ 0 do
8:             d ← depth
9:             r ← red
10:            S_Key[r][d] ← CompS_key(Key)[r][d]
11:            TgServAd[r][d] ← CompAd(S_Key[r][d])
12:            TgServ ← GetTg(TgServAd[r][d])
13:            if route(Query, TgServ) then
14:                i++
15:                put(SID, Query)
16:            end if
17:            d−−
18:        end while
19:    end for
20:    return 0
21: end function
```

These mechanisms enable our cloud system to cope with a non-uniform growth of the system and they ensure that a service will be stored in a redundant way that will maximize the success rate of its retrieval. Our solution has the property of consistent hashing: if one resource server (or a service of the resource server) fails, only its keys are lost, but the other binders are not impacted and the whole cloud system remains coherent. As in many existing systems, pairs will be stored by following a hybrid soft and the hard state strategy.

A pair (service name, address) will have to be stored by its creator every x period of time, otherwise it will be flushed by the binders that store it. A delete message may be sent by the creator to remove the pair before the end of the period. Algorithm 4 describes services storage's process on the servers y using Service IDentifier.

When a user wants to use a service, it is connected to any resource server on the cloud and chosen the service name's. Next, current service server's, by using previous mechanism hashes name and finds associated service by sending retrieving query in the cloud. Retrieving query is processed and different servers where is stored the service are located. Then, it is possible to execute one of following operation (Algorithm 3):

- Deleting: deleting operation allows to make unavailable all the occurrences of a service through replication in the system,
- Updating: it allows to bring changes in a service through all these replications,
- Inserting: this allows to add a new service to the the cloud system (i.e., in all resources servers of replication).

Altogether, the mechanisms of storage and lookup that we propose are (on one hand) reliable, in the sense that we always manage to store or to find any service in our system of cloud. On the other hand, this system is strong because it assures an availability of the services in a strongly dynamic environment.

VI. Simulations

In this section, we present the results of the simulations that we have performed and we have assessed the practicability, and in some cases the scalability, of our addressing. Our cloud system is considered as dynamic (the phenomenon of churn is applied). We use the Peersim [39] simulator for cloud computing system simulation and it allows to service name following the uniform distribution. Our simulation involves the following parameters of the DHT used in our cloud (Excepted, the study on the floating point precision issue). These parameters are valid for all the DHTs that we compare:

- Number of servers of services connected and used to store different services is equal to 10000 in the starting up, maximum number of services by server equal to 2000.
- We consider that our cloud system is dynamic and the rate of churn varying from 10% to 60%.
- We consider a simulation performed during 2 hours.
- The leaving and joining of the servers of services follow an exponential distribution as well as the inserting and the deleting of the services of the clouds associated.
- We suppose that the system receives 6 millions of queries following an exponential distribution with a median equal to 10 minutes.

A. Characteristics of our hyperbolic tree

1) ***Analysis of the angular gap in the Poincaré disk****:* In this part, we try to analyse the average gap between points computed on the unity's circle.

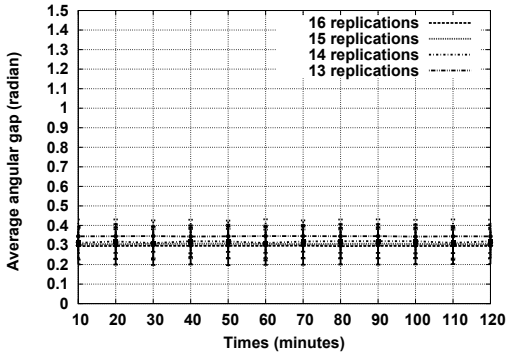

Figure 7. Angular gap from 13 to 16 circular replications.

This study will allow us to see the level of efficiency of our strategy of circular replication. Indeed, according to the number of chosen circular replications, in our case 16, there is a threshold for which the probability two primary binders is confused.

Except, if two binders at least are confused, it means reducing of number of replication actually made by the system

and of the same cost to reduce the rate of the storage and lookup queries.

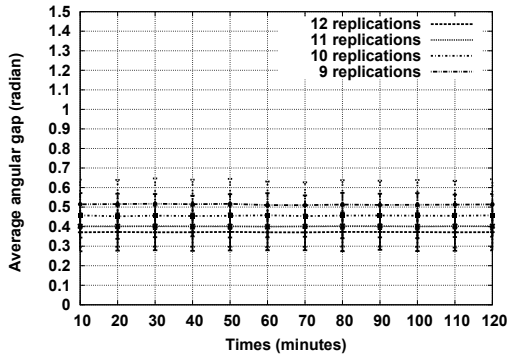

Figure 8. Angular gap from 9 to 12 circular replications.

Figures 7-10 show in the context of our simulation the average gaps between points calculated of the circle unity as well as their standard deviations.

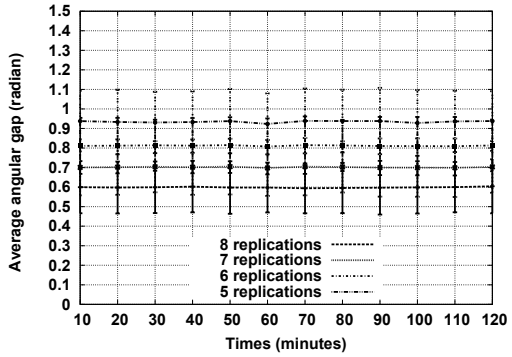

Figure 9. Angular gap from 5 to 8 circular replications.

In a general way for all the figures, we notice a growth of the average value of the angular gaps, which is inversely proportional at the level of used replication. Indeed, we have a variation of 0.29 for 15 replications in approximately 1.4 out of 2 replications.

This indicates that our system behaves well because even for a large number of circular replications, the angular gap is enough important so that we obtain (in most of the time) different primary binders. Furthermore, for example, we can notice that in Figure 7, which concerns the average angular gaps in levels of replication between 16 and 13, the values remain constant during all the period of simulation.

So, we have gaps between 0.29 and 0.35 with standard deviations about 30% of the average. This indicates that approximately 78% of the calculated points have an angular gap between 0.20 and 0.40 for 16 replications and between 0.24 and 0.46 for 13 replications. So, in Figures 7-10 the averages seem to be significant because, in each case, we

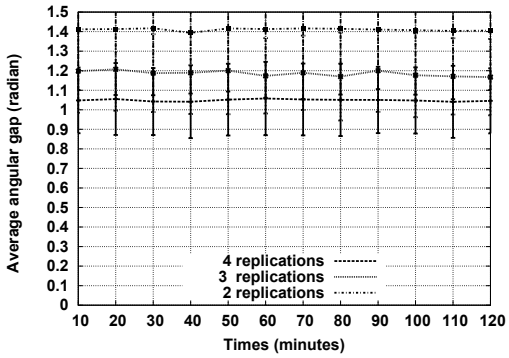

Figure 10. Angular gap from 2 to 4 circular replications.

observe a standard deviation at the most equal to 30% of average.

These results tend to confirm that our strategy of circular replication is reliable in the measure or at a given level of replication, we can associate various primary binders, so avoiding that a departure of stacker not in the loss of information of a large number of nodes.

*2) **Analysis of the number of sub-keys**:* Figure 11 presents the evolution of the number of primary binders according to the number of sub-keys chooses for the simulation.

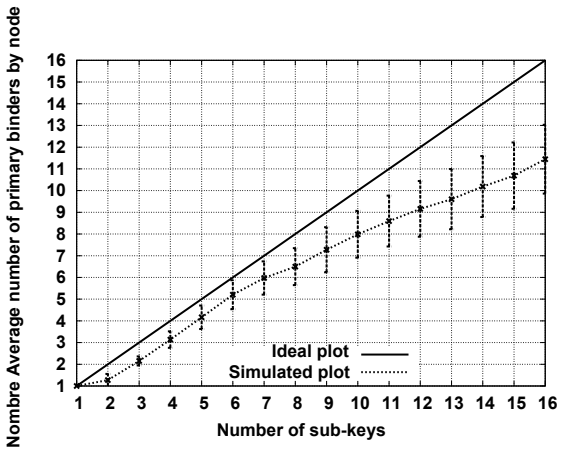

Figure 11. Variation of binder number depending of the number of sub-keys

This curve shows a continuous growth of the number of binders according to the number of money keys. Furthermore, we can observe that this curve is below the ideal case or the number of binders always corresponds among key money. The feigned curve is very close to the ideal curve, which shows that we have a satisfactory situation for all the levels of replication.

B. Load balancing in the cloud

Figure 12 shows an experimental distribution of points corresponding to the scatter plot of the distribution of servers

of services in our cloud. Thus, we can mark that hyperbolic-tree of our cloud system is balanced. Indeed, we can noticed by part and others around the unit circle, which we have servers of services.

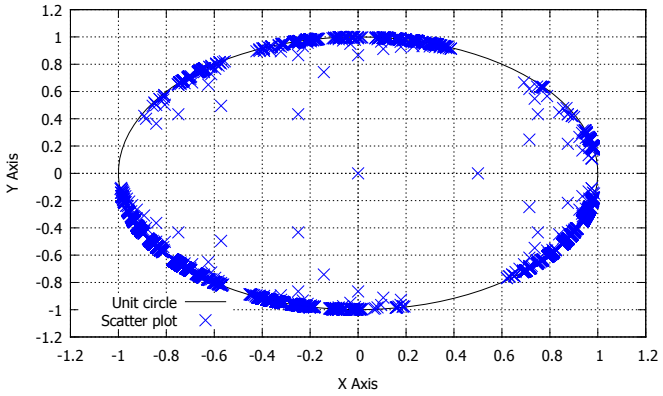

Figure 12. Scatter plot corresponding to the distributed database servers.

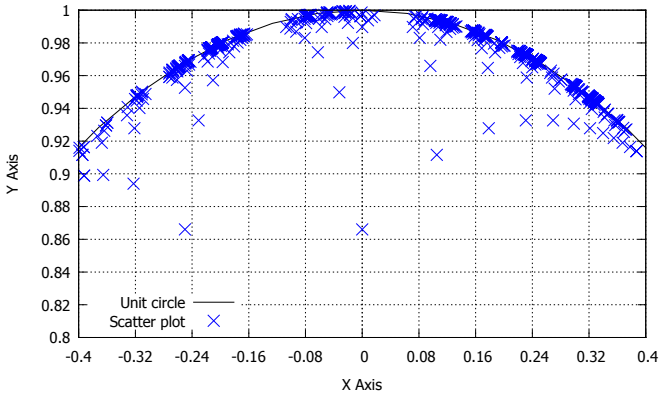

Figure 13. Distribution of nodes in the neighbourhood of the edge of the unit circle.

This has an almost uniform distribution around the root, which implies that our system builds a well-balanced tree what will more easily allow to reach a load balancing of storage.

Figure 13 shows correspondingly Poincaré disk model that no address of resource server belongs on the edge of the unit circle. Indeed, the addresses of resource server were obtained by projection of the tree of the hyperbolic plane in a circle of the Euclidean plane of radius 1 and of centre with coordinates (0;0).

C. Floating point precision issue

One property of the Poincaré model is misleading: the distances are not preserved. If we observe the Poincaré model from an outside point of view, the distance are smaller than the reality (i.e., inside the plane). Because of the model is a representation of the hyperbolic plane in Euclidean plane. Indeed, closer to the boundary of the circle are the points, the farther they are in reality. This phenomenon is illustrated in Figure 13 obtained by simulation.

The hyperbolic plane has a boundary circle at infinity represented in the Poincaré unit disk model (i.e., the open unit disk) by a circle of radius 1 and centred on the origin O (Section III). The open unit disk around O is the set of points whose the complex modulus is less than 1: $|w| < 1$ with $|w| = \sqrt{(w_{Re})^2 + (w_{Im})^2}$.

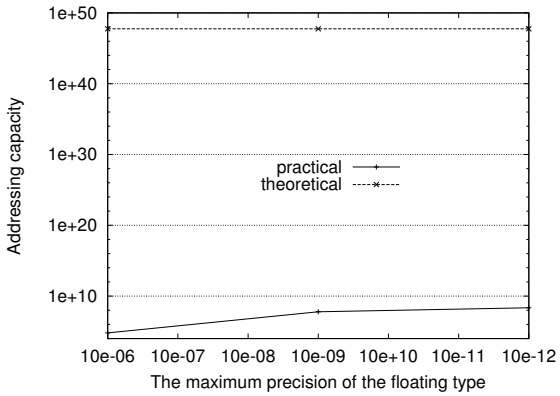

Figure 14. Addressing capacity as a function of the floating point precision threshold.

In practice, an embedding of such a mathematical space is constrained by the precision of the floating type used, typically a *double*. It is a problem of arithmetic precision, we reach the maximal accuracy allowed by the computing in floating point. Indeed, the calculations obey with the IEEE 754 standard, which determines the binary floating point representation. The floating point arithmetic can be implemented with variable length significant that are sized depending on the needs. This is called Arbitrary Precision Arithmetic (APA). To compute with extended precision we have found three computational solutions: computation with rounding (e.g., IEEE 754), interval arithmetic and the Real Ram model.

But we should use a specific library such as the *MultiPrecision Complex* MPC library. As the complexity of using APA is important and as we have enough addressing capacity by using standard floating point numbers, we keep on using the *double* type representation. Thus, two points cannot be closer that the minimum non zero *double*. Hence, the minimal half space is the space that can contain one distinct point.

To address this issue we should answer at the question of Paul Zimmermann: Can we compute on the computer? In order to assess the impact of overlay parameters such as the degree and the depth, we carry out significant simulations, we try to find the best tree parameters to maximize the addressing capacity. This brings us to some practical concerns:

- How to determine the maximum number of subspaces that we can create to assign a coordinate to a node?
- What is the maximal node density in a subspace?

To carry out our practical analysis, we proceed as follows. We embed a tree with a degree of 32 and a depth equal to 32. Then, we assign an address to each node. We show in Figure 14 the gap between the number of addresses in theory and in practice. We set the maximum precision to a given value and compute the addresses. We then vary the maximum precision with the significant digits evolving from 6 to 12 (i.e., 10e-6 to 10e-12). With these characteristics, the theoretical addressing capacity is the same whatever the precision, namely

maximum (as expected). The addressing capacity increases strongly between an accuracy of 6 to 9 digits compared to the transition of 9 to 12 digits because more disjoint points appear.

1) ***Influence of the degree of hyperbolic tree****:* Finally, for a degree different of 32, the addressing capacity stays closer to 2.246E+08.

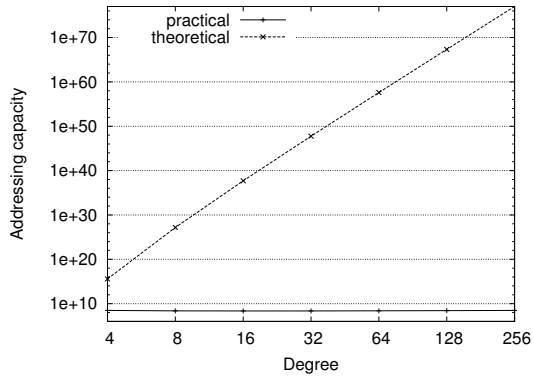

Figure 15. Influence of the degree on the number of theoretical addresses.

To analyse the influence of the degree on the addressing capacity, we use a precision of 12 digits and a tree depth of 32 hops. The tree degree evolves from 4 to 256. Figure 15 shows that the theoretical addressing capacity increases linearly in function of the degree, unlike the "practical" plot that seems constant. In fact, we show in Figure 16 that with a degree higher than 32 the gain is weak compared to the order of magnitude observed in Figure 15.

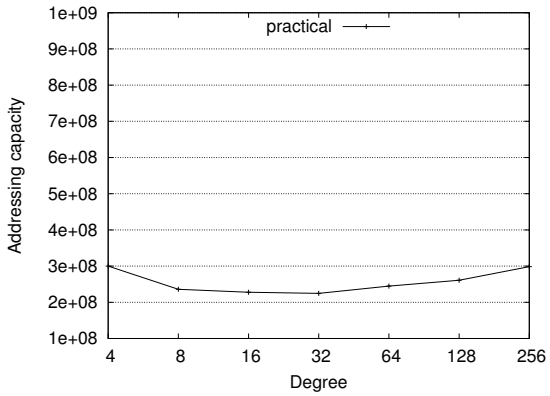

Figure 16. Influence of the degree on the number of practical addresses.

2) ***Influence of the depth****:* In the same way, we analyse the influence of the depth on the addressing capacity. The precision is the same as the previous one and the tree degree is set to 32. The tree depth evolves from 4 to 32. In Figure 17, the increase of the theoretical addressing capacity is exponential when the depth increases. As expected, this matches with the normal characteristics of $\mathbb{H}^2$. On the other side, in practice, the addressing capacity achieves the threshold at 2.246E+08.

The threshold is reached with only a depth of 8. Indeed, the boundary of the disk is quickly reached. We recall that the

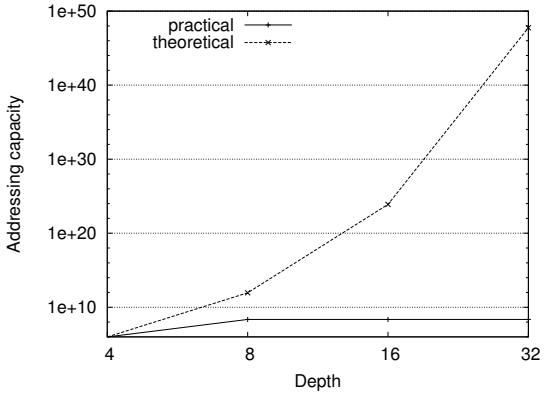

Figure 17. Influence of the depth on the number of addresses.

tiling is built with regular $q - gon$, q is the number of sides of the $q - gon$. In $\mathbb{H}^2$, whatever q, there exists a distance d from which the created subspaces are pair wise disjoints (i.e., the sides of the q-gon are tangent as the red line in Figure 3). This property is more explained in Section IV. Because of this distance and the precision of the floating type (12 digits), the leaves of the tree can reach the boundary of the disk after only 7 hops (i.e., a depth of 7).

A fine tuning of the degree parameter can improve the addressing capacity, namely we can set q to the degree of the tree that we find the most suitable. This is possible because we create an overlay network, we can have some freedom in setting the overlay links and thus we can restrain the degree of the addressing tree.

D. Performances analysis

1) ***Evaluation of the substitution strategy****:* Figure 18 shows us the impact of the substitution strategy on the average number of binders reached by every node during the process of storage. Indeed, we compare two situations.

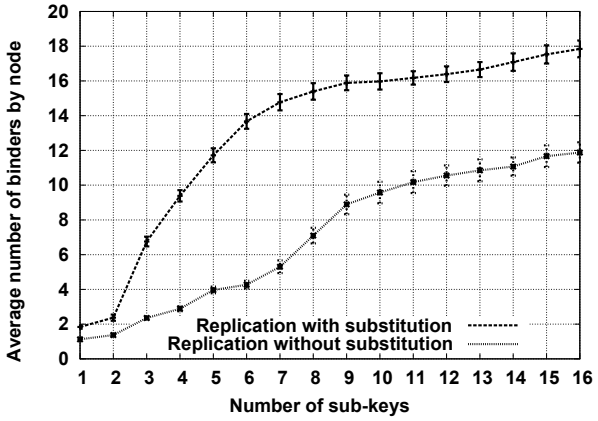

Figure 18. Evaluation of the performances of the substitution strategy

At first, the case where the new resources servers join in a random way the system via the first resource server having free address. Then, the case or the new resources servers try to substitute themselves for binders having left the network to continue to assume this role.

In Figure 18, we notice that according to the number of sub-keys used, average numbers of binders reached in the case of the substitution are much greater than in the case of a connection of new nodes to the first random address. This established fact shows that the substitution strategy returns us a stronger system in the sense that the couples keys-values of the various nodes are distributed on a large number of binders so returning the available information even in case of departure of binders. So, for example, by using 8 sub-keys, we have approximately 7 binders on average during all the simulation in the strategy of circular replication and simple radial road against about 14 binders when we use the substitution method besides the classic replication. Furthermore, for 16 sub-keys, we have approximately 12 binders against about 18 on average in the case of the substitution.

2) ***Success rate with the substitution strategy****:* Figure 19 presents two plots illustrating respectively the evolution of the rate of success of the storage queries in the case of a classic replication (circular and radial) then in the case of a classic replication with which we associated the substitution strategy.

We can easily note that the level of classic replication, the success rate offered by the substitution strategy is better. Indeed, in a replication, we note a rate of average success about 62% of successes for the classic replication against about 75% of the method of substitution.

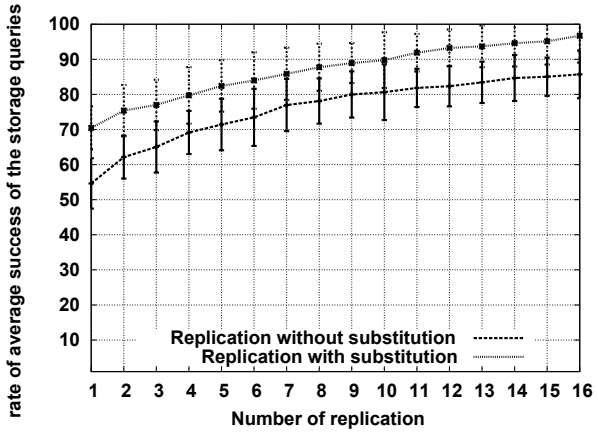

Figure 19. Evaluation of the success rate in storage context

In 7 replications, we note 78% of successes in the case without substitution against 89% in the case using this strategy.

With 15 replications, we observe 85% of success rates in without substitution against 97% of successes in the case of the substitution. Generally, we can notice on average an earnings of about 10% in the success of the storage queries.

These results show the interest in using the strategy of replication, which constitutes an additional factor in the improvement of the success of the storage queries of services (Figure 19). The same strategy impacts positively services lookup, as shown in Figure 20.

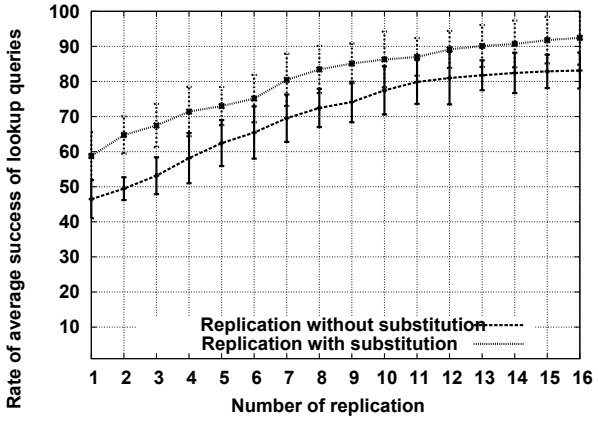

Figure 20. Evaluation of the success rate in lookup context

In the case of queries for a lookup, the situation is almost similar because, as we can observe in Figure 20, a better rate success when we use the substitution strategy. Indeed, also in it an earnings of an average 9% of successes on queries for a lookup. Whether it is for the storage or for the lookup, we can say that our strategy is reliable because it allows us to increase considerably the success rates.

VII. Conclusion

In this paper, we have proposed a new Cloud platform, providing scalable, reliable and robutess distribution of services. Indeed, these properties are an essential requirement for this kind of system. There were few research reports proposed for multi-dimensional indexing schemes for Cloud platform to manage the huge and variety services to address the complex queries efficiently (because most of these proposed systems do not have properties given behind). In this study, we have showed that our solution using hyperbolic-tree with virtual coordinates is consistent, reliable, scalable, and hardy, because it supports enough well churn phenomenon. Furthermore, through the study of floating point precision issue, we showed the limits of the capacity of addressing of our hyperbolic tree depending to the degree and to the depth. Thus, our theoretical analysis of the Poincaré disk model has shown that even though we lose a huge number of potential addresses because of the floating point accuracy limits, we can still handle a vast amount of servers of services (i.e., 100M order of magnitude) before reaching those limits of the model. Our system has properties, which allow to guarantee the availability of the Cloud services. Furthermore, it gives very good results when we apply the substitution strategy in churn context. In the future work, we would like to investigate the impact of churn phenomenon issue and thus to develop a new multi-dimensional index structure, which ensures a best result in dynamic context in the aim to supply a variety of cloud services.

REFERENCES

[1] T. Tiendrebeogo, "A new spatial database management system using an hyperbolic tree," in DBKDA 2015 : The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications, May 2015, pp. 43–50.

[2] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications." New York, NY, USA: ACM, 2001, pp. 149–160.

[3] M. Castro, M. Costa, and A. I. T. Rowstron, "Performance and dependability of structured peer-to-peer overlays," in International Conference on Dependable Systems and Networks (DSN), 28 June - 1 July 2004, Florence, Italy, Proceedings, 2004, pp. 9–18.

[4] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric." London, UK, UK: Springer-Verlag, 2002, pp. 53–65.

[5] "Cloud computing model," August, 2015, https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/Nuage33.png/400px-Nuage33.png.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep., Feb 2009.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Commun. ACM, vol. 53, no. 4, Apr. 2010.

[8] C. Baun, M. Kunze, J. Nimis, and S. Tai, Cloud Computing: Web-Based Dynamic IT Services, 1st ed. Springer Publishing Company, Incorporated, 2011.

[9] P. Murray, "Enterprise grade cloud computing," in Proceedings of the Third Workshop on Dependable Distributed Data Management, ser. WDDM '09. New York, NY, USA: ACM, 2009, pp. 1–1.

[10] A. Weiss, "Computing in the clouds," netWorker, vol. 11, no. 4, Dec. 2007.

[11] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," Georgia Institute of Technology, Tech. Rep., 2009.

[12] T. Jowitt, "Four out of five enterprises giving cloud a try." Computerworld UK (visited: 2010, May 7), 2009. [Online]. Available: Available:http://www.computerworlduk.com/management/it-business/services-sourcing/news/index.cfm?newsId=16355

[13] R. L. Grossman, "The case for cloud computing," IT Professional, vol. 11, no. 2, Mar. 2009.

[14] A. Plesser, "Four out of five enterprises giving cloud a try," Tech. Rep., 2008. [Online]. Available: Available:http://www.computerworlduk.com/management/it-business/services-sourcing/news/index.cfm?newsId=16355

[15] B. Huang and Y. Peng, "An efficient two-level bitmap index for cloud data management," in Proceedings of the 3rd IEEE International Conference on Communication Software and Networks (ICCSN). IEEE, 2011, pp. 509–513.

[16] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The characteristics of cloud computing," in Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, Sept 2010, pp. 275–279.

[17] "Cloud services model," August, 2015, https://www.simple-talk.com/cloud/development/a-comprehensive-introduction-to-cloud-computing/.

[18] S. Zhang, S. Zhang, X. Chen, and S. Wu, "Analysis and research of cloud computing system instance," in Proceedings of the 2010 Second International Conference on Future Networks, ser. ICFN '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 88–92. [Online]. Available: http://dx.doi.org/10.1109/ICFN.2010.60

[19] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, ser. MobiCom '03. New York, NY, USA: ACM, 2003, pp. 96–108. [Online]. Available: http://doi.acm.org/10.1145/938985.938996

[20] D. Hilley, "Cloud computing: A taxonomy of platform and infrastructure-level offerings," Tech. Rep., 2009.

[21] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," Gaithersburg, MD, United States, Tech. Rep., 2011.

[22] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: A viable solution?" in Proceedings of the 2008 International Workshop on Data-aware Distributed Computing, ser. DADC '08. New York, NY, USA: ACM, 2008, pp. 55–64. [Online]. Available: http://doi.acm.org/10.1145/1383519.1383526

[23] Z. Pervez, A. M. Khattak, S. Lee, and Y.-K. Lee, "Csmc: Chord based session management framework for software as a service cloud," in Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, ser. ICUIMC '11. New York, NY, USA: ACM, 2011, pp. 30:1–30:8. [Online]. Available: http://doi.acm.org/10.1145/1968613.1968650

[24] J.-Y. Wu, J.-l. Zhang, T. Wang, and Q.-l. Shen, "Study on redundant strategies in peer to peer cloud storage systems." Applied Mathematics Information Sciences, 2011, pp. 235–242.

[25] D. S. and K. Piromsopa, "Research on cloud storage environment file system performance optimization," vol. 2. Information Management, Innovation Management and Industrial Engineering (ICIII), 2011, pp. 58–62.

[26] C. Miguel, C. Manuel, and R. Antony, "Performance and dependability of structured peer-to-peer overlays." in proceedings of the 2004 DSN (IEEE), 2004, pp. 9–18.

[27] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," vol. 22, 2004, pp. 41–53.

[28] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in Revised Papers from the First International Workshop on Peer-to-Peer Systems, ser. IPTPS '01. London, UK, UK: Springer-Verlag, 2002, pp. 53–65. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687801

[29] N. Antonopoulos, J. Salter, and R. M. A. Peel, "A multi-ring method for efficient multi-dimensional data lookup in p2p networks." in FCS, H. R. Arabnia and R. Joshua, Eds. CSREA Press, 2005, pp. 10–16. [Online]. Available: http://dblp.uni-trier.de/db/conf/fcs/fcs2005.html#AntonopoulosSP05

[30] X. Zhang, J. Ai, Z. Wang, J. Lu, and X. Meng, "An efficient multi-dimensional index for cloud data management," in Proceedings of the First International Workshop on Cloud Data Management, ser. CloudDB '09. New York, NY, USA: ACM, 2009, pp. 17–24. [Online]. Available: http://doi.acm.org/10.1145/1651263.1651267

[31] J. Wang, S. Wu, H. Gao, J. Li, and B. C. Ooi, "Indexing multi-dimensional data in a cloud system," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 591–602. [Online]. Available: http://doi.acm.org/10.1145/1807167.1807232

[32] S. Wu and K.-L. Wu, "An indexing framework for efficient retrieval on the cloud." vol. 32, no. 1, 2009, pp. 75–82. [Online]. Available: http://dblp.uni-trier.de/db/journals/debu/debu32.html#WuW09

[33] A. F. Beardon and D. Minda, "The hyperbolic metric and geometric function theory," in International Workshop on Quasiconformal Mappings And Their Applications, 2006, pp. 9–56.

[34] R. Kleinberg, "Geographic routing using hyperbolic space," in in Proceedings of the 26th Annual Joint Conference of INFOCOM. IEEE Computer and Communications Societies, 2007, pp. 1902–1909.

[35] D. Krioukov, F. Papadopoulos, M. Boguñá, and A. Vahdat, "Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces," SIGMETRICS Perform. Eval. Rev., vol. 37, no. 2, 2009, pp. 15–17.

[36] A. Miquel, "Un afficheur générique d'arbres à l'aide de la géométrie hyperbolique," in Journées francophones des langages applicatifs (JFLA), 2000, pp. 49–62.

[37] F. Papadopoulos, D. Krioukov, M. Boguñá, and A. Vahdat, "Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces," in Proceedings of the 29th Conference on Information Communications, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 2973–2981. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833893

[38] V. Gaede and O. Günther, "Multidimensional access methods," ACM Comput. Surv., vol. 30, no. 2, Jun. 1998, pp. 170–231. [Online]. Available: http://doi.acm.org/10.1145/280277.280279

[39] I. Kazmi and S. F. Y. Bukhari, "Peersim: An efficient & scalable testbed for heterogeneous cluster-based p2p network protocols." Washington,