A Cloud Computing System Using Virtual Hyperbolic Coordinates for Services Distribution

Telesphore Tiendrebeogo(✉)

Polytechnic University of Bobo, Bobo-dioulasso, Burkina Faso
tetiendreb@gmail.com

Abstract. Cloud computing technologies have attracted considerable interest in recent years. Thus, these latters became inescapable in most part of the developments of applications. It constitutes a new mode of use and of offer of IT resources in general. Such resources can be used "on demand" by anybody who has access to the internet. Cloud architecture allows today to provide a number of services to the software and database developers among others remote. But for most of the existing systems, the quality of service in term of services' indexation is not present. Efforts are to be noted as for the search for the performance on the subject. In this paper, we define a new cloud computing architecture based on a Distributed Hash Table (DHT) and design a prototype system. Next, we perform and evaluate our cloud computing indexing structure based on a hyperbolic tree using virtual coordinates taken in the hyperbolic plane. We show through our experimental results that we compare with others clouds systems to show our solution ensures consistence and scalability for Cloud platform.

Keywords: Virtual coordinates · Cloud · Hyperbolic plane · Storage · Scalability · Consistency

1 Introduction

The deployment of Cloud Computing in our recent everyday life has strongly to modify the perception which we have of the notion of software, working platform as well as infrastructure subjected to licenses. Cloud Computing constitutes a commercial solution with a bright future. Indeed, it concerns most part of the services used in companies, going of the value for financial interesting for the acquisition of software services to the compromise between the energy consumption by the servers and the on-line acquisition of the storage spaces. Cloud Computing constitutes a system of virtual computation with the possibility of maintaining it and of managing it at a remotely. From a structural point of view, he can be characterized by the following aspects:

Y. Zhang et al. (Eds.): CloudComp 2015, LNICST 167, pp. 269–279, 2016.
DOI: 10.1007/978-3-319-38904-2_28

- IaaS (Infrastructure as a Service).
- PaaS (Platform as a Service).
- SaaS (Software as a Service).

In this paper we make the following contributions:

- We introduce a new technique of virtualisation based on the Poincaré disk model in which a q-regular hyperbolic tree is used to model the resources through its various nodes;
- We show how the indexation of the various resources is made through a greedy routing algorithm of the requests. [6];
- We show the properties of scalability and of consistence in term of indexation;
- We perform some simulations and we show that our cloud system using based on a structured DHT is comparable and sometimes better than others structures based on existing index structures, such as Chord, MSPastry, Kademlia with possibility to run multi-attribute criteria request and multi-dimensional indexation.

The continuation of the paper is organized as follows. Section 2 provides a brief overview on the related works in the indexation in Cloud Computing. Sect. 3 presents the properties of the hyperbolic plan used in Poincaré disk model. Section 4 defines the local addressing and greedy routing algorithms of cloud computing system. Section 5 describes the mechanism of addressing and the technique of greedy routing in the hyperbolic tree. Section 6 makes an analysis of the results of the simulation of our model of Cloud Computing and we conclude in Sect. 7.

2 Related Work

We can distinguish various types of system as Distributed storage for management of big quantity of data, such as Google File System [7] (GFS), which serves Google's applications with an important volume of data. Yahoo provided PNUTS [8], a hosted, centrally controlled parallel and distributed database system for Yahoo's applications. These systems, split data and constitute some fragments, then disseminates randomly these latters into clusters to improve data access parallelism. Some central servers working as routers are responsible of the queries orientation to nodes which contain query results. Unlike these works, we propose a scalable mechanism using Poincaré disk model and which provides distributed data storage and retrieving algorithms based on the in hyperbolic space. Our indexation structure is designed to route by greedy way a big quantity of queries among a large cluster of storage nodes by hyperbolic coordinates using. Consistent hashing proposed by the previous works is designed to support key-based data retrieval but is not adapted to support range and multi-dimensional queries. It exists some solutions that support query processing over multi-dimensional data, like CAN (Content Addressable Network) [9]. It permits to build a database storage system by splitting rectangular areas.

Furthermore, a lot structured DHT algorithms (MSPastry [12], Tapestry [17], Kademlia [13], CAN, and Chord [11]) that support multi-dimensional range queries may be used to implement some cloud services such as Chord based Session Management Framework for Software as a Service Cloud (CSMC) [14], MingCloud based on Kademlia algorithm [15], Search on the Cloud is built on Pastry [16] and improves fault-tolerance, scalability and consistence.

Our work is associated to the design proposed in [18]. However, to be able to answer the multi-dimensional queries, we propose a new algorithm which uses the mechanism of greedy routing for the process data storage and data lookup. Furthermore, our structure reduces considerably the number of hops for the storage and the lookup inside of the Cloud system, so facilitating the deployment of databases for the applications.

3 Poincaré Disk Models Properties

The initial model of the hyperbolic plane that we will consider is due to the French mathematician Henri Poincaré. This model is called, Poincaré Disk Model. In this latter, the hyperbolic plane is represented by the open unit disk of radius 1 centered at the origin (coordinates associated to the origin is (0;0)). In this specific model:

- All the points are located inside of the open unit disk.
- Lines correspond to arcs of a circle intersecting the disk and meeting its boundaries at right angles.

In the Poincaré disk model, every point is identified by complex coordinates. One of the important properties of the hyperbolic plan is that we can tile by using polygons of any size, called called p-gons. Each tessellation is represented by a set {p, q} where every polygon has p faces with q of them in every vertex. The values p and q so presented obey the following relation: $(p-2)*(q-2) > 4$. In a tiling, p is associated to the number of sides of the polygons of the *primal* (indicated in black vertices and blue edges: Fig. 1) and q correspond to the number of sides of the polygons of the *dual* (indicated by the red triangles: Fig. 1). Our purpose is to split the hyperbolic plane in the aim to give an unique address to each node. We set p to infinity, thus transforming the primal into a q-regular tree. The dual is then tessellated with an infinite number of q-gons. In this way, we arrive has to create an embedded tree in the hyperbolic plan by splitting of plan into tessellation which we use to address system nodes. An example of such a hyperbolic tree with $q = 3$ is shown in Fig. 1.

The distances between any two points u and v in the Poincaré disk model are given by curves minimizing the distance between these two points. These distances are called geodesics of the hyperbolic plane. The value of a geodesic between two points u and v is represented by $d_{\mathbb{H}}$, The Poincaré metric considered as an isometric invariant is given by following relation:

$$d_{\mathbb{H}}(u,v) = argcosh(1 + 2 \times \frac{|u-v|^2}{(1-|u|^2)(1-|v|^2)}) \quad (1)$$

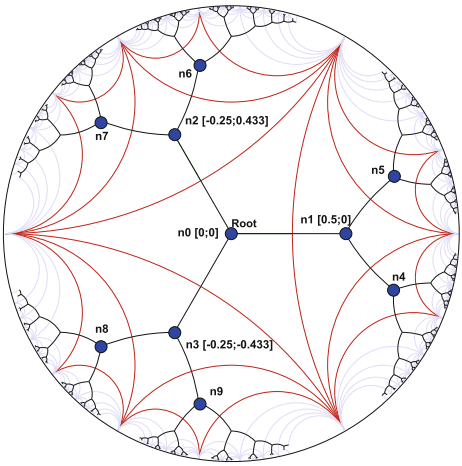

Fig. 1. 3-regular hyperbolic tree of Poincaré disk model.

This formula is used by the greedy routing algorithm shown in the next section.

4 Hyperbolic Greedy Routing

In this section we present the principle of hyperbolic addressing tree building on one hand and on the other hand, we show how various resources servers of Cloud communicate through queries. We propose in this paper a dynamic and scalable algorithm for routing's process based on greedy routing algorithm mechanism. In the initial phase, a first resource server is started and defined the number (q)of resources servers in the which it can connect (the degree of the tree). With the aim of being able to identify the various nodes of the tree associated to the resources servers, we use complex coordinates (taken in the hyperbolic plan). Each node of the hyperbolic tree has q possibilities to connect others nodes, called children of current node. The degree corresponds to addressing capacity of each resource server. The building strategy of cloud is incremental, with each new node resource joining one or more existing resources servers. This method is scalable because unlike [1], we do not have to make a two-pass algorithm over the whole cloud system to find its highest degree. In our cloud system, a node can connect to any other node at any time in the aim to obtain coordinates. The initial phase 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 tilling at the origin of the disk in function of q. The principle of splitting of the space in two separate sub-space is assured to be unique if both half-space are tangent; hence the *primal* is an infinite q-regular tree. We use the theoretical infinite q-regular tree to built 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 Fig. 1). Furthermore, each node 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 calculate the coordiantes that can be given to these children. The first node takes the hyperbolic address (0;0) and is the root of the tree.

Algorithm 1. Calculating the coordinates of a nodes's children.

```
1: procedure CalcChildrenCoords(node, q)
2:     step ← argcosh(1/sin(π/q))
3:     angle ← 2π/q
4:     childCoords ← node.Coords
5:     for i ← 1, q do
6:         ChildrenCoords.rotaLeft(angle)
7:         ChildrenCoords.translat(step)
8:         ChildrenCoords.rotaRight(π)
9:         if ChildrenCoords ≠ node.ParentCoords then
10:            StoreChildrenCoords(ChildrenCoords)
11:        end if
12:    end for
13: end procedure
```

Our distributed algorithm ensures that the nodes are contained in distinct spaces and have unique coordinates. All the phases of the presented algorithm are suitable for distributed and asynchronous computation. Thus, it allows the assignment of addresses as coordinates in dynamic topologies. Each node can obtain an address by asking a node already connected to system. The node supplying the address so becomes the parent of the new node. Therefore, the knowledge global of the system is not necessary. Each node wishing to connect to the system asks for an address a node of the system. If the node has not it, the query is routed in the direction of another node. Each time that node want to connects to the system, it computes its hyperbolic coordiantes of it future children. When a new node is connected to the cloud, it share these resources with others resources servers associated to the nodes of the cloud, by sending queries. The routing process from source to destination is done by step by using the greedy Algorithm 2 based on the hyperbolic distances between the nodes.

In a real context of cloud, link and node failures are expected to happen often. Indeed, if the addressing tree is broken by the failure of a node or link, we flush the addresses attributed to the nodes beyond the failed peer or link and reassign new addresses to those nodes (some nodes may have first to reconnect with other nodes in order to restore connectivity).

Algorithm 2. Routing a query in the cloud.

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

5 Strategy of Naming and Binding on Our Cloud System

We approach on this part the strategy consisting in taking the name of a server of resources then has to transform it into address with the aim of facilitating the data storage and the data lookup (this address corresponds to the virtual coordinates who allows to locate the resources server). Our solution uses a structured Distributed Hash Table (DHT) system that with the virtual addressing mechanism of resources servers associated to the greedy routing algorithms presented in Sect. 4. At the beginning, each new entity (resources server) takes a name that is associated to the service (Application, Platform, Infrastructure) that it wishes to share in the system. The name in question is kept by the entity during all its life cycle in the system. When a resources server connects to the system having obtained an address, it begins the process of storage of the various services which wishes to share on other resources servers. This storage uses a mechanism of fragmentation in sub-key of the key obtained by hashing of the entity name(as explain in the follow). When a similar sub-key is already stored in the system, an error message is generated and sent back to the resources server containing concerned service in the aim to change the service identifies. Thus, Unity of the service name is assured.

For each node is associated the pair (name, address), with the name mapping as a key is called a *binding*. Figure 2 shows the way every binding is stored in the cloud. A binder is associated to the any entity that stores these pairs. The depth corresponds among hops from given node towards the root by following the direct relationship links (including the root itself). When the cloud system is created, the system chosen a maximum depth associated to the potential binders. Thus, the depth allows to compute the maximum number of entities that can be connected to the system and potentialy share different services. Also the depth d choice must verify d that minimize the Inequality 2 with p ($p \geq 3$) corresponding to the degree:

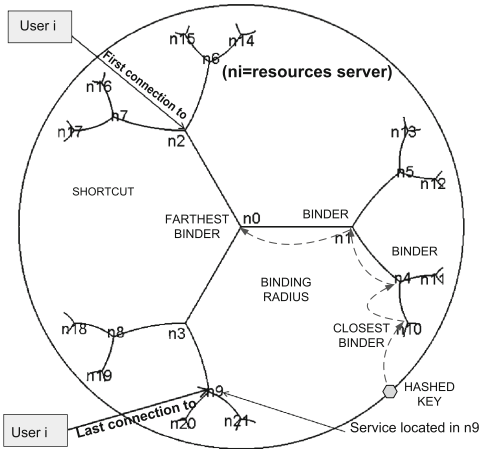

Fig. 2. Hyperbolic cloud system.

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

This value is defined as the *binding tree depth*. When a new entity joins the system by connecting to other entities, it obtains a virtual coordinates from one of these entities. So, each service name of the resources server is transformed into a key by hashing its identifier with the SHA-1 algorithm (SHA-1 gives 160-bit key). Next, the new entity divides the 160-bit key into 5 equally sized 32-bit sub-keys (for redundant storage). One sub-key is randomly selected to be transformed into angle by use of a linear function given then. The angle is given by:

$$\alpha = 2\pi \times \frac{\texttt{32-bit sub-key}}{\texttt{0xFFFFFFFF}} \tag{3}$$

Once the obtained angle, the entity computes the virtual point v situated on the edge circle unity

$$v(x,y) \text{ with} \begin{cases} x = cos(\alpha) \\ y = sin(\alpha) \end{cases} \tag{4}$$

Next the enity identifies the locations of the closest binder to the computed virtual point above by using the given *binding tree depth*. In Fig. 2 we set the *binding tree depth* to three to avoid cluttering the figure. We have to notice that the closest entity of the circle unity can not exist. Indeed, the closest address can not have been to request by an entity. In which case, the query is redirected step by step towards the next node which contains the service either towards the root (use of the greedy algorithm of the Sect. 4). In the general way, this process continues until the query reaches to the root entity having the address (0;0) (which is the farthest binder) or the number of entities is equal to (radial strategy):

$$S \leq \lfloor \frac{1}{2} \times \frac{log(N)}{log(q)} \rfloor \tag{5}$$

with N equal to number of entities or distributed resources servers, q to degree of hyperbolic-tree.

To reduce the impact of the dynamics of the system (departures and arrived from node in the system), our system uses the redundancy mechanism which allows to guarantee a certain robustness of the cloud in the difficult contexts. This redundancy mechanism allows to store several copies of a service on a number of nodes at the radial level (according to the value of the replication radial chosen arbitrary) then at the circular level (according to the value of the circular replication chosen arbitrary).

These mechanisms entrainnent about can a not uniform growth of the system. On the other hand, they assure a bigger probability to be able to lookup or to store a service. It so maximizes the rate of success of lookup and the storages and participle in the flexibility and in the availability of the services. Our solution has the property of consistent hashing. Indeed, if one entity (or a service of the entity) fails, only its keys are lost, but the other binders are not concerned and the whole cloud system remains coherent. To avoid storing a service in a server that is going to leave later the system without the latter updates this departure. Our system requires that periodically (x this period) pair (name, address) is again stored in the system.

When a user i wants to use a service, it connects to the system and enters the name of the service. The latter is then hashed in key and splits into sub-key before to be send in the system as the lookup key. This lookup key will allow to locate the servers being able to provide this service in the cloud.

6 Experimental Results

In this section we are to focus on the results of the simulation of a model of cloud which we implemented on Peersim simulator [21] to analyze the scalability and the availability of the services. Our configuration of simulation takes into account the phenomenon of churn which shows the dynamics of the system. In our configuration, the services are uniformly distributed (i.e. each service name that is randomly generated preserves equi-probability in term of storage in the servers). For the simulation, we used the following parameters that are valid for all the DHTs that we compare:

- at beginning, number of resources servers connected and which is share services is equal to 10000;
- dynamics factors varies between 10 % and 60 % include;
- duration of simulation is equal in 2 hours;
- exponential probability law is used to qualify servers or services churn effects;
- total number of queries supposed received by our system is equal to 6 millions of queries following exponential probability law with median equal to 10 min;
- for each server we have a maximum number of services equal to 2000.

6.1 Load Balancing in Our System

Figure 3 shows the dispersal points of the cloud corresponding to the location of the various services servers in our hyperbolic addressing tree. We can easily noticed that our tree seems well-balanced. We can note that most part of nodes finds itself around the unit circle and distributed in a fair way. This implies that our builts system is balanced well and allows to realize a load balancing.

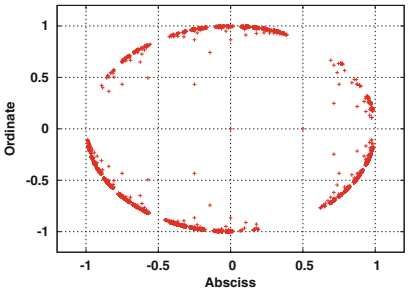

Fig. 3. Scatter plot of our system entities.

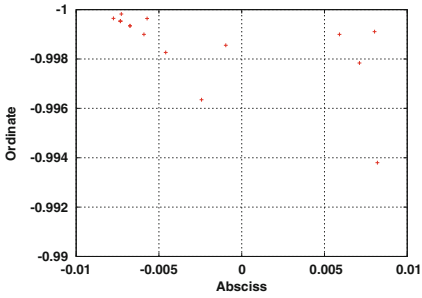

Fig. 4. Distribution of various entities in the neighborhood of the edge of the unit circle.

Figure 4 offers more precision as for the distribution of the entities around the edge of the unit circle. Indeed, more generally about is the number of entities of the system, they are all contained in the Poincaré disk and verify the relation $1 = cos^2\alpha + sin^2\alpha$.

6.2 Performances Analysis

Figure 5, shows that the rate success varies between 83 % and 88 % when the phenomenon of churn varies between 10 % and 60 %. Furthermore, in the absence

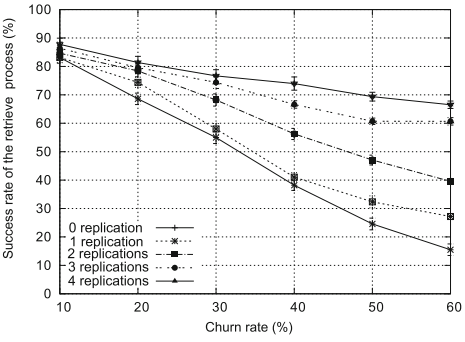

Fig. 5. Impact of the replication on the phenomenon of churn.

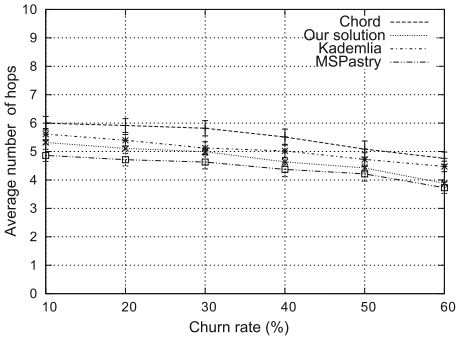

Fig. 6. Comparative analysis between different DHTs.

of replication, rate success varies between 18 % and 67 %. This result indicates that the replication has the effect of improving the rate of success of the services lookup in our cloud.

Figure 6, indicates The results obtained in term of rate of success when churn varies between 10 % and 60 % according to the various DHTs are appreciably the same. We show through this experience that our system is comparable to the existing cloud systems based on the existing DHTs such as Chord, MSPastry and Kademlia.

7 Conclusion

In this paper we propose a system of cloud which supplied scalability, flexibility and availability of the services. Very few search results proposes architecture of cloud with the which is associated a technique of muti-dimensional indexation as our. Our cloud model exploited the properties of the Poincaré disk and allows thanks to its technique of hashing it can develop strategies of replication. We showed by simulation how our system resists to the churn phenomenon. In our future works, we are going to emphasize the elaboration of a servers substitution technique in the cases of failures of these latters, in the aim to improve the rates of success of the queries. Furthermore during our future works, we consider aborderles mechanism of safeties partners in the discovery of services on our cloud

References

1. Kleinberg, R.: Geographic routing using hyperbolic space. In: IEEE Computer and Communications Societies (2007)
2. Zhang, S., Zhang, S., Chen, X., Wu, S.: Analysis and research of cloud computing system instance. In: Proceedings of the Second International Conference on Future Networks, pp. 88–92 (2010)
3. Antonin, G.: R-trees : A dynamic index structure for spatial searching. SIGMOD Rec. **14**(2), 47–57 (1984)
4. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database System Implementation. Prentice-Hall, Upper Saddle River (2000)
5. Wang, J., Wu, S., Gao, H., Li , J., Chin, O.B.: Indexing multi'-dimensional data in a cloud system. In: SIGMOD International Conference on Management of Data. ACM (2010)
6. Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., Stoica, I.: Geographic routing without location information. In: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking. ACM (2003)
7. Vaquero, L.M., Luis, R.-M., Juan, C., Lindner, M.: A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput. Commun. Rev. **39**(1), 50–55 (2008)
8. Cooper, B.F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.A., Puz, N., Weaver, D., Yerneni, R.: PNUTS: Yahoo!'s hosted data serving platform. In: VLDB Endowment (2008)
9. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-addressable Network. ACM, New York (2001)

10. Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: An Efficient multi-dimensional index for cloud data management. In: CloudDB 2009. ACM (2009)
11. Antonopoulos, N., Salter, J., Peel, R.M.A.: A multi-ring method for efficient multi-dimensional data lookup in P2P networks. In: FCS. CSREA Press (2005)
12. Castro, M., Costa, M., Rowstron, A.: Performance and dependability of structured peer-to-peer overlays. In: Proceedings of the DSN 2004 (2003)
13. Hou, X., Cao, Y., Zhang, Y.: P2P Multi-dimensional range query system based on kademlia. Comput. Eng. **20**, 14 (2008)
14. Zeeshan, P., Masood, K.A., Sungyoung, L., Lee, Y.K.: CSMC: chord based session management framework for software as a service cloud. In: ICUIMC 2011. ACM (2011)
15. Ji-yi, W., Jian-lin, Z., Tong, W., Qian-li, S.: Study on redundant strategies in peer to peer cloud storage systems. Appl. Math. Inf. Sci. **5**(2), 235S–242S (2011)
16. Savage, R., Nava, D.T., Chàvez, N.E., Savage, N.S.: Search on the cloud file system. In: PDCS: Parallel and Distributed Computing and Systems Conference (2011)
17. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: a resilient global-scale overlay for service deployment. IEEE J. Sel. Areas Commun. **22**(1), 41–53 (2004)
18. Wu, S., Wu, K.L.: An indexing framework for efficient retrieval on the cloud. IEEE Data Eng. Bull. **32**(1), 75–82 (2009)
19. Lu, J., Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: CIKM. ACM (2003)
20. Ion, S., Robert, M., David, K., Frans, K.M., Hari, B.: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Comput. Commun. Rev. **31**, 149–160 (2001)
21. Kazmi, I., Bukhari, S.F.Y.: PeerSim: An efficient & scalable testbed for heterogeneous cluster-based P2P network protocols. In: UkSim 2011. IEEE Computer Society (2011)