2011 International Conference on Complex, Intelligent, and Software Intensive Systems

Using self-led Critical Friend topology based on P2P Chord algorithm for node localization within Cloud Communities

Stelios Sotiriadis[1], Nik Bessis[1,2], and Nick Antonopoulos[1]
[1]School of Computing & Maths, University of Derby, UK
[2]Department of Computer Science and Technology, University of Bedfordshire, United Kingdom
stelios@sotiriadis.gr, (n.bessis, n.antonopoulos)@derby.ac.uk

Abstract — Resource provision within critical friend environment take place on a demand fashion and it is based on the aptitude of members to look across multiple locations for resource discovery and allocation. A major concern of such large scale and uncertain topology setting is the capability of members (nodes) to efficiently search and locate neighbouring participants. Here we adopt a Peer to Peer (P2P) approach in which every node in the network acts alike and discovers resources in a distributed coordination. More specifically, we propose the use of Chord algorithms, as a distributed peer to peer lookup protocol, put forward a new solution by assigning keys to different nodes. By specifying the aspect in which keys (data) are assigned to nodes, and how a node can determine the value for a given key, the algorithm locates the node responsible for that key. In the work herein we discuss a notable case, namely how Chord algorithms as a distributed P2P lookup protocol may determine provable solutions to the problem of efficient large scale Grid and Cloud resource localization. More specifically, the aforementioned proposal deals with the load balancing, scalability and availability of Grid and Cloud resources in a decentralized manner.

Keyword: Cloud Computing, Grid Computing, Self-led Critical Friends community model, P2P searching protocols, Chords algorithms

I. INTRODUCTION

Grid is defined as a coordinated endeavour of numerous nodes, normally connected in random topologies, in order to reach a common scientific or technical goal. Set of those resources with common characteristics, such as familiar administrative sharing rules and actions are referring as a Virtual Organisation (VO). The VO is composed of a set of entities (e.g., resources, services, and users) from different autonomous domains collaborating in order to complete some cross-organization cooperative tasks [12]. In parallel, over the recent years, the notion of Clouds has proven to be a successful commercialized model, which encompasses Grid computing while moves it one step forward. By incorporating Grid and Utility computing; Cloud aims to eliminate the over-provisioning of resources and offers an on demand resource synthesis environment.

Literature suggest that a Cloud needs thin nodes; members which functionality depends profoundly on some other administrative node to fulfil its traditional computational role, as well as Grid and Utility computing (a shared server usage cost) [1].However, the resource capability of a single Cloud is generally limited, and some applications often require various Cloud centres over Internet to deliver services together. Current trends are tending to shift Cloud functionality to a more decentralized manner. This yields thin clients (in our case the nodes) to self-directed behaviours which based upon their own perception regarding to the whole knowledge domain. We call these neighbouring of nodes as Critical Friends Community (CFC) and each specific node as a Self-led Critical Friend (SCF) [3]. Moreover SCF comprises the role of mediator in the communication by reflecting inter-connections to any inter-connected node. In this study we aim to utilize the aforementioned CFC nodes and we deal with an efficient resource discovery and provisioning algorithm of a Grid environment (part of the Cloud) in which members will be capable of performing self-sufficient actions and locate neighbouring nodes more powerfully.

With the purpose of addressing the associated objectives, we focus on the Cloud part and how the aforementioned goal may be achieved using a P2P Chord algorithm. P2P systems and applications are distributed environments of nodes without a centralized control and hierarchical structure. Each node performs and manages an action or functionality in alike to all other nodes. By reviewing all the P2P features such as redundant storage, permanence, selection of nearby servers, anonymity, search, authentication, and hierarchical naming [4], we conclude that the core operation among those rich characteristics is the localization of nodes. For this research study, the Chord algorithm is selected with the aim to associate its specific functionality of the distributed hash table's in order to efficient lookup nodes. The distributed hash table (DHT) puts forward a global view of data items and consistently maps nodes and their data items into a common address space. The great advantages of DHT are that search queries for data items or nodes are routed via a small and predetermined number of nodes to the target node. It offers a load balancing mechanism with an equally handling means among all nodes for retrieving their data. Finally a DHT is robust against random failures [7].

In the next sections, we discuss the motivation of the study (Section 2), and definition of similar technologies (Section 3). The rest of the paper outlines the Critical Friends P2P Chord algorithm (Section 4, 5) as well an applicable searching Chord protocol (Section 6). At last, we conclude with the future work section and the proposed challenges part (Section 7, 8).

978-0-7695-4373-4/11 $26.00 © 2011 IEEE
DOI 10.1109/CISIS.2011.80

IEEE computer society

II. STATE OF THE ART

During the last years, evolvement of Cloud has produced various visions of their resource provision topology models. A widely accepted vision is the cluster configuration of a homogeneous workload of a node which is controlled by the same local resource management system. Nowadays, a new scheme gets the attention of the academic community; the fully decentralized models which turn clusters of collaborated enterprises into an efficient community of cluster members. The major problem within such settings is the resource discovery and scheduling of jobs to any linked nodes. Their interactions are based upon their knowledge of the domain. For that reason a solution of self-sufficient member may be a personalized profile of internal data. Nodes snapshot profile may include: known and trusted nodes addresses, hardware and software data, previous works delegation records, policies, etc. Utilizing those profiles joining to and leaving from the neighbouring of nodes happens dynamically without affecting the whole functionality of the VO domain. This may offer the prospects towards to multi-institutional and unrestrained Clouds.

These multi-tenancy environments of nodes include interconnections with other VOs participants by composing an extended and scalable environment. [2, 3] discuss the neighbouring of nodes as CFC composed by inter-collaborated member knows as *Self-led Critical Friend* (SCF). By acting as mediators in the communication they reflect communication to any trusted node. On a similar vein, [6] addressed a notable case namely how a SCFs topology should be the means to realize interoperability and clarifies that a grid community can communicate within their VOs, thus they can form and manage their own perceptions about neighbouring nodes based on previous interactions, such as communication and delegation records. In other words, by using SCF, the discovery of nodes is based on a nodes internal knowledge independent of its VO domain.

The major problems that developers have to face within such environments are issues related to scalability and reliability. A decentralized distributed network of members should not be affected by a single point of failure and manage newly added and unused resources. In this work we present a P2P case study of a Chord algorithm. Nodes in our case contribute to the resource discovery as providers and consumers of services by acting autonomously. Members of the CFC collaborate directly with each other by utilizing information of their snapshot profile by consistently hashing addresses using a DHT model.

Several authors discus the great benefits of P2P in the area of decentralization of resources. The first generation of the P2P networks contains Napster, Gnutella and Freenet. Napster allows nodes to communicate to each other though a central index server which collects and sends queries directly to clients (nodes). Despite the fact that Napster can ensure the correct results, there is always the setback of a single point failure (centralized management) as well as the bottleneck of scalability. An alternative solution is the Gnutella project which contains a decentralized search index mode unlike to Napster. A node asks for data of interests from neighbouring nodes and this process continues so on. Although the major problem here is the flooding of queries, search is actually distributed but not scalable. Finally, Freenet which has been designed to provide anonymity uses "smart queries" which data flows are in reverse path of query. Despite Freenet offers a good performance within small scale networks, communication messages must be passed over many hops, in order to search through the system to find the data. Each hop not only adds to the total bandwidth load but also increases the time needed to perform a query. Those drawbacks drive developers to the next generation P2P settings by approaching them from a different perspective by introducing DHT.

The second generation of P2P networks is the structured architecture in which nodes are self-organised with load balancing and fault tolerance mechanisms. Their main functionality based on a DHT interface. A DHT stores key and value pairs in which keys are similar to a file name and values are the file content. Their main target is to efficiently insert, lookup and delete key and value pairs. Each peer (in our case node) stores a subset of key-value pairs. The core operation is to find node responsible for that key by mapping a key to node. Their great strengths are that keys are mapped equally to nodes while each node maintains information for only a small number of other nodes. The DHT is a generic interface and there are several implementations of it. The most well known are the Chord, Pastry, and Content Addressable Networks (CAN) [5].

Chord organises nodes in a circle based on node identifiers. The keys are assigned each time to the successor node within the circle. Each node has a finger table with a specific size using a finger table function. The searching method lookups in finger table for the furthest node that precedes the key. Pastry architecture is similar to Chord. More specifically, Pastry considers network locality to minimize hops messages travel. The new nodes need to know a neighbouring node to achieve locality. Finally CAN architectural design is "a virtual multi-dimensional Cartesian coordinate space, a type of overlay network, on a multi-torus" [9]. The nodes within the space are identified with coordinates. "The entire coordinate space is dynamically partitioned among all the nodes in the system such that every node possesses at least one distinct zone within the overall space" [9].

While there is some scepticism about the potentials of P2P networks not only as a file sharing mechanism but also as a powerful model in several computing areas (e.g. Cloud computing), we believe that their rapid wide-spread deployment of unstructured P2P networks may offer significant advantages. In this study we use the Chord algorithm method in order to structure a specific VO, and we deal with CFC nodes localization. By describing their functionality we aim to discuss a large scale combinatorial search with high probability of finding accurate solutions.

III. THE CHORD P2P ALGORITHM

Chord is designed to offer high flexibility while may be implemented by general purpose systems [7]. For achieving resource localization it uses a consistent hashing distributed

lookup system [7] for referring keys to nodes. The consistent hashing function assigns each node and keys an identifier using Secure Hashing Standard (SHA-1) base hash function. By providing a unique mapping among nodes and an identifier space, each node is recognized by an IP address and a port number (Chord identifier) which is hashed to a unique identifier. The Chord architecture includes the Chord as the means to map identifiers to nodes and the DHT that associates nodes identifiers to values.

Nodes are placed in particular positions within a Ring (Chord). Equally the nodes pair keys and values are placed in the same Chord. In order to implement the hash function, peers contain an *m-bit* identifier for both nodes and keys. Each key has a successor and a predecessor. The *m-bit* identifiers are stored within an m-entry routing table namely as the finger table. As *m* we denote a sufficient big number that make collisions improbable. This table stores information about other nodes participating in the neighbouring (Each table record consists of a node identifier and a network address). Together key identifiers: SHA-1(key) and node identifier: SHA-1(IP address) are uniformly distributed within the same space.

Nodes' identifiers are arranged in the Chord according to the function $mod2^m$. The successor of a node called *successor (k)* and is the first node whose identifier is greater of the k in the identifier space. The theorem of Chord algorithm is that for or any set of nodes *n* and keys *k* with high probability:

- Each node is responsible for at most (1+e)k/n node keys.
- When $(n+1)^{st}$ node joins or leaves the domain the responsibility for *O(k/n)* keys changes to other nodes.
- Each node finger table holds unique *m* entries (the number of bits in identifier) in the finger table up to the number given by the function *O(log n)*.
- The i^{th} entry in the table at a node *n* contains the identity of the first node *s* that succeeds *n* by at least 2^{i-1} on the identifier circle. So, the succesor $s = successor(n+2^{i-1})$.

Figure 1 illustrates the successor discovery algorithm as discussed previously.

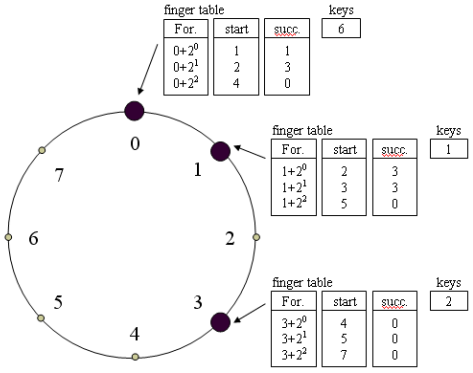

Figure 1: Finding nodes' successor

In the above example finger tables shows the successors of nodes in which for node 0 are $s1_{node0}= 0+2^0=1$, $s2_{node0}= 0+2^0=1$ etc. Finally, following this procedure Chord nodes identify their successors and predecessors.

IV. The Chord Cloud Community Protocol

The Chord Cloud Community protocol described below utilizes the Chord P2P algorithm in order to achieve the following objectives:

- Create the first Chord Ring topology utilizing the initial VO members. The system uses an initial bootstrap node.
- Hashing the metadata snapshot profile information, the IP address and Port as also the physical resources of node.
- Insert a new node to the Chord which allocates a specific position.
- Insert a new key with snapshot profile within the Chord.
- Delete an existing node in the case of communication failures.
- Delete an existing key (snapshot profile).
- Search for a specific or a set of keys (snapshot profiles).

In the next sections we describe the aforementioned objectives from the CFC perspective. We assume that different VOs may cooperate with each other in order to achieve an extended environment. As consequent we originally created a Bootstrap Chord Ring (BCR) in which the initial nodes (already existed) are registered and take their places within the Chord.

Starting from scratch, nodes belonging to several VOs are scheduled to be placed in the ring using the BCR functionality. The *m-bit* identifier for each node is generated by using a SHA-1 function. The snapshot profile is hashing as well as the node IP in order to create the key hash. Each hash key placed at the appropriate location in the Chord. For instance, a node with key *20* will be placed to a position with key value equal or greater than *20*. It should be mentioned that, the SHA-1 creates m size hash keys with 160 bit, so it is unlikely to have two equal identifiers of keys. Consistent hashing places the nodes to a circle of modulo 2^m. The key *k* goes to the first node which key is equal or less, making this node the successor of k, successor(k). In a clockwise topology identifiers are placed precisely in the Chord as numbers from 0 to 2^{m-1}. Figure 2 demonstrates CFC VOs in which members are allowed to be inter-connected with nodes belonging to different VOs. Each member contains a snapshot profile with useful information such as CPU power, total memory, hard disk space and numbers and a number of constraints regarding their VO (security, policies etc).

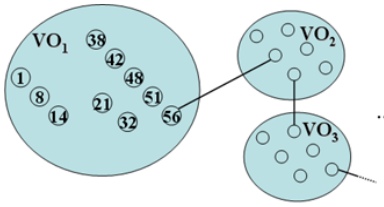

Figure 2: The VO topology

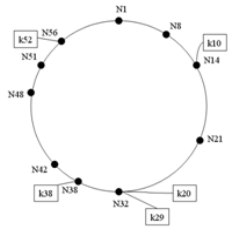

Figure 3: The Chord Ring

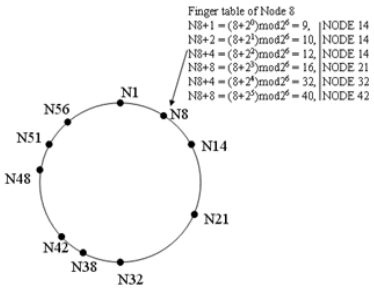

Figure 4: The finger table

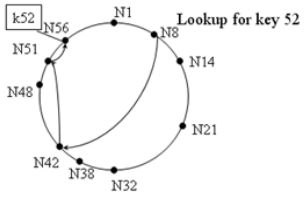

Figure 5: Lookup for key 52

In our sample a VO1 contains *10* nodes and *5* keys are stored within the Chord with information of snapshot profiles. Each node is linked to a successor and predecessor node. Figure 3 shows the topology of the Chord. As illustrated above the successor of k10 is the N14, so the key is stored to the node14. The same procedure happens with keys k20, k29 goes to node 32, k38 to node N38 and k52 to Node N56. A Chord load balancing mechanism ensures that keys are spread equally to nodes. Each time a new key enter to the Chord the (1+e)K/N where e=O(logN) function ensures the highest number of keys that nodes are responsible for.

Chord offers two lookup methods for searching specific keys. The simple lookup protocol based upon the successor functionality. So, when a node requests for a key asks its successor which performs a serialized forward to the next one and so on until the desired key found. Despite the fact that this solution is provable to find the correct results, it is time consuming [7]. For solving that, Chord offers a finger table; a DHT which stores information for a specific number of nodes. More specifically, each node *n* holds a DHT with *m* records which is called finger table. The i^{th} record of the finger table contains the identifier of the first node s which is the next one of node n at least 2^{i-1} positions in the circle. So the successors are given by the function successor($n+2^{i-1}$), where $1 \leq i \leq m$. This node is called the i^{th} finger of node n. Each record of the finger table contains the Chord identifier and the IP address (with the port). So the next finger of node *n* is the successor (next node) of *n* within the circle.

Figure 4 shows that the finger table values of node8. The first finger of node8 is node 14 as $N8+1 = (8+2^0)\text{mod}2^6 = 9$ and the next are calculated evenly. This technique offers two great advantages. Firstly each node contains a small number of nodes which are located as nearest to the node as possible. Secondly the key searched procedure works faster than before as forwards passing from tables to tables. Figure 5 shows the lookup function for key 52. The node 8 searches for the successor of key 52. The finger table first redirect to node42 as 52 is greater than the last entry of the table. Then node42 forwards search to node 51 as 42 is less or equal to 51. Finally 51 discovers that node 56 is the responsible node.

V. JOINING OF NEW NODES TO CHORD

Herein we discuss how Chord deals with new connections and possible node failures. Before starting a key search, nodes may change their status; therefore Chord must ensure that successors are stored correctly. This comes up by utilizing a stabilization protocol which runs periodically in the background and refreshes information of the finger tables. When a new node n is activated for first time, calls either the join function, or the create function in order to build a new Chord (using a bootstrap node). The join function doesn't update the rest of the nodes with information of a new node which is about to join. In order to achieve it the Chord uses the stabilization method which essentially discovers the new nodes connected to the Chord. Each time a node runs the stabilization function, decides which the successor for the predecessor p is. This case could happen when a new node p joins the Chord. Furthermore, the stabilization function updates the successor of node n that n is still active by giving the opportunity to store n as a predecessor. Every node runs periodically the finger fix function in order to make sure that records of the finger table are updated. Using this method, a node initializes its finger table as well as store new nodes. Additionally, a checking for predecessor procedure should occur, which will verify periodically that fingers to predecessors are the correct.

VI. SCFS CHORD SEARCHING PROTOCOL

This section introduces the Community Chord Protocol which aims to incorporate different Cloud VOs to a single domain. Essentially, using the Chord P2P protocol we attempt to specify how to locate nodes, how to add new nodes and how to recover from the failure or planned departure of existing nodes. The general aim is to describe the way in which a) VOs are generated to Chords and b) the inter-collaboration model is transforming to inter-Chord cooperation. Then we discuss in c) the procedure of joining a new node (a CFC member) to the inter-Chord partnership and d) how peers (nodes) search for specific information (keys). Finally in e) we describe the node failures occasions. The protocol is organised as follows:

A. The Chords Generation

Let us assume that a VO_a contains *i* nodes which are able to communicate with each other based upon their own perceptions about their acting domain, so they form a community. Each member of this community contains a profile with data about neighbouring nodes (local IP address,

neighbouring addresses of previous job delegations, policies, physical information etc.). The algorithm first transforms their local IP address (and a port) to an m-bit identifier using the SHA-1 hash function. The key for our search is the desired physical resources. The algorithm places the nodes to the Chord. For *n* number of VOs the algorithm generates *n* number of Chords. At this time each node is assigned with a finger table showing its successor nodes. This procedure is demonstrated at Figure 6; nodes from VOs are placed Chord Rings (Layer 1 to Layer 2)

B. The inter-collaborated VOs

In aforementioned sections we have discussed that CFC members are actually links between different VOs in order to extend their current boundaries. Let assume that among two VOs called VO_a and VO_b, at least two nodes are existing which are inter-connected forming a collaborative environment. Their functionality is to perform one single operation; they redirect communication to other members belonging to the same VO. In other words, when a searching for keys is initialized those members will pass queries to their trusted peers. Those peers will not be aware of which node requests for queries and in which VO this query has been generated. We assume that VO_a size is greater than VO_b size, so we place the inter-connected node of VO_b to the VO_a Chord. At this time, we have two interconnected nodes. In our sample the VO_b node is placed at the $Chord_a$ and the Chord joining procedure starts. This procedure is showing at Figure 7 (see Layer 2).

C. Joining the Chord

The new node is added to the $Chord_a$ at a location according to its m-bit identifier. After that, new nodes successors and predecessors are considered. Affected nodes, runs periodically the stabilization protocol in order to update finger tables and successor protocols. It should be mentioned, that the new position of the newly added node is decided by the Chord. The new node now belongs to both $Chord_a$ and $Chord_b$ so it is possible to redirect searching queries to both VOs. This theorem is possible as Chords theory confirm that if any sequence of join operations is executed interleaved with stabilizations, then at some time after the last join the successor pointers will form a cycle on all nodes in the network [4]. The procedure is demonstrated in Figure 7 (Layer3).

D. Searching for keys

We assume that a node from VO_a searches for a specific key. The node will forward requests to their Chord finger nodes (nodes from finger table). This procedure will continue until the desired key is found. During this procedure, the newly added node-belonging to both $Chord_a$ and $Chord_b$-may forward requests to the incorporated Chords. With high probability, the number of nodes that must be contacted to find a successor in a N-node network is O(log N). The total performance will not be affected, because the Chord theorem suggests that if we take a stable network with N nodes with correct finger pointers, and another set of up to N nodes joins the network, and all successor pointers are correct, then lookups will still take O(log N) time with high probability [4].

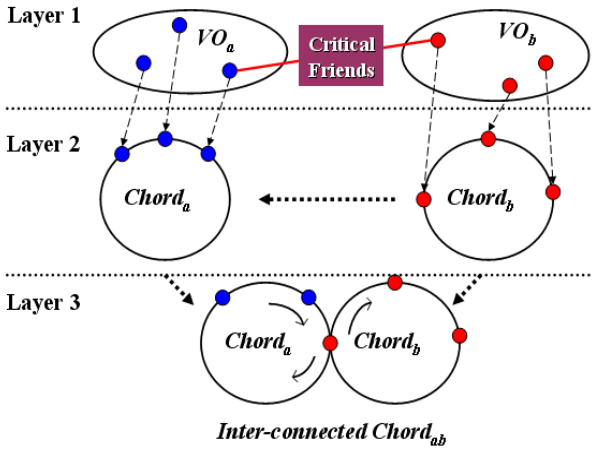

Figure 6: CFC Chord protocol

E. Node failures

The key mechanism to deal with nodes failures is to maintain correct successor pointers. To achieve this, each node maintains a *successor-list* of its *r* near successors on the Chord. In the case of a failure, node *n* notices that its successor has failed, it replaces with the first live entry in the list. The *stabilize* method will correct finger table entries and successor-list entries pointing to failed node. Performance is sensitive to the frequency of node joins and leaves versus the frequency at which the stabilization protocol is invoked [8].

VII. Theoretic Evaluation of the Model

The aforementioned model aims to the development of an inter-Chord resource provisioning model. We treat members as CFC nodes, which communicate with each other based upon their own perceptions as regards the domain knowledge. The concept is straightforward; we are aspired from the P2P Chord protocol in order to achieve parallel objectives, searching for resources within structured and flexible VOs. At a first glance, a parallelism of both technologies shows that major advantages of Chord network such as fault-tolerance, scalability, self-organisation, decentralisation and effective lookup are dependable requirements for a successful Cloud setting.

We first start our method by transforming a Cloud into a Chord. Each member IP address (along port) is hashed using the SHA-1. Then nodes are placed into the Chord according to their hash id which we have called node identifier. The snapshot profile is also hashed and keys are spread across the Chord members in a way in which the key id is greater or equal to the node identifier. The same procedure occurs for every VO. At the same time we assume that nodes from different VOs may communicate with each other forming a CFC neighbouring. The inter-Chord collaboration model initializes inter-cooperated members which always join the Chord with the smallest number of nodes. Then a Chord stabilize method will run which will transform Chords according a pattern. The pattern will be able to decide new Chords size and their inter-connections.

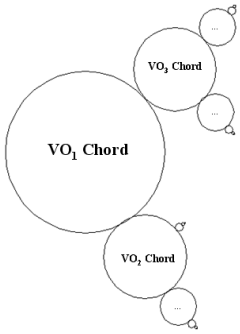

Figure 7: The inter-Chord generations

Figure 7 illustrates that procedure. Using this method member we can achieve efficient resource lookup as Chord network can find a data or send a message to any arbitrary node in only O(log N) steps.

VIII. Inspiration Of the Inter-Chords Relations Pattern

To facilitate management and control of the inter-Chords communication it is required to develop a pattern algorithm. This pattern has to be able to manage and organise an infinite number of Chords in a structured way. More specifically, we suggest that arrangement and construction of inter-Chords may be correlated to the Mandelbrot set. If Chords are treated as fractals we may design a complex and infinite scale of VOs. Fractals are objects that display self-similarity at various scales. Magnifying a fractal reveals small-scale details similar to the large-scale characteristics [10]. Although the Mandelbrot set is self-similar at magnified scales, the small scale details are not identical to the whole. In our method, we first create an initial population of inter-chords and continue generation at the same pattern. In fact, the pattern of inter-Chords connections have to be similar to the Mandelbrot set.

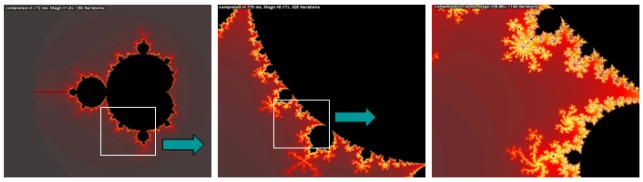

Figure 8: The Mandelbrot Set pattern

The Mandelbrot set is a connected set of points in the complex plane. For achieving that we need to pick a point z_0 in the complex plane and calculate next points as: $z_1=z02+zo$, $z2=z12+z0$, $z3=z22+z0$, etc. If the distance of the sequence is 2 continually, then point z0 is said to be in the Mandelbrot set. According to [11], figure 8 shows that by zooming in on the first image shows out self similarities in the basic pattern and so on.

IX. Conclusion

In the work herein we have addressed a notable opportunity to see Cloud as a Chord of members by utilizing the notion of the CFC model as a way to achieve the inter-cooperation of distributed nodes. Using this simple and powerful protocol, we have suggested that Cloud nodes could maintain information about O(log N) of other nodes. The lookup protocol also needs the same amount of messages in order to efficiently locate the desired data (key). The mechanism scales sufficient with the increased number of nodes and continues to function well despite the major changes may happen in the system.

X. References

[1] Schubertt, L., Jeffery, K., and Neidecker-Lutz, B., Expert Group Report, The future of Cloud Computing: Opportunities for European cloud computing beyond 2010, European commision, Belgium, 2010, Available at: http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf, Accessed at: 10/10/2010.

[2] Sotiriadis, S., Bessis, N., Huang, Y., Sant, P., Maple, C., "Defining minimum requirements of inter-collaborated nodes by measuring the heaviness of node interactions". in: International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010), IEEE, Krakow, Poland, February 2010.

[3] Huang, Y., Bessis, N., Brocco, A.., Sotiriadis, S., Courant, M., Kuonen, P., Hisbrunner, B., "Towards an integrated vision across inter-cooperative grid virtual organizations". in: Future Generation Information Technology (FGIT 2009), pp.120-128, Springer LNCS, Jeju island, Korea, 2009.

[4] Brunskill E., 2001. Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service. In Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HOTOS '01). IEEE Computer Society, Washington, DC, USA,

[5] Tsoumakos D., and Roussopoulos N., 2006. Analysis and comparison of P2P search methods. In *Proceedings of the 1st international conference on Scalable information systems* (InfoScale '06). ACM, New York, NY, USA

[6] Bessis, N., Huang, Y., Norrington, P., Brown, A., Kuonen, P., And Hirsbrunner, B., 2010. Modelling of a Self-led Critical Friend Topology in Inter-cooperative Grid Communities, International Journal of Simulation Modelling Practice and Theory, Elsevier, (available online on July 7, 2010, to appear in Volume 19, Issue 1, January 2011), ISSN: 1569-190X, p.p.: 5-16.

[7] Dan, Y., XinMeng, C., and YunLei, C., 2005. An Improved P2P Model Based on Chord. In Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT '05). IEEE Computer Society, Washington, DC, USA, 807-811.

[8] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.,R., Kaashoek, M., F., Dabek, F., and Balakrishnan, H., 2003. Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans. Netw,11, 1 (February 2003)

[9] Rahnama , A., Habibi, J., Rostami, H., Abolhassani H., 2005, A semantic addressable network, IADIS International Conference on WWW/Internet, p.p 43-51.

[10] Sun, N., Miyazaki, R., and Yoshida. N., 2010. Complex mapping with the interpolated Julia set and Mandelbrot set. In ACM SIGGRAPH ASIA 2010 Posters (SA '10). ACM, New York, NY, USA, , Article 49

[11] Schmidt, H., Mandelbrot Applet: http://www.h-schmidt.net/MandelApplet/mandelapplet.html, Accessed at 22/1/2010

[12] Li, J., Li, B., Du, Z., and Meng, L., CloudVO: Building a Secure Virtual Organization for Multiple Clouds Collaboration. In Proceedings of the 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '10). IEEE Computer Society, Washington, DC, USA