Cloud Tree: A Hierarchical Organization as a Platform for Cloud Computing

Khaled A. Nagaty

***Abstract*—Applications using current cloud organization suffer from high rate time latency which results in long response time which makes real time applications unable to meet real-time constraints. Also, these applications suffer from a very low coverage ratio of their clients. Human activities such as cooperation, collaboration, making decisions and entertainment are significantly affected. In order to reduce time latency on the cloud and increase clients' coverage ratio we propose a modification to the current cloud organization to become hierarchical. This hierarchy includes an artificial organizational entity called a master datacenter at the top with domain and backend datacenters as descendents. Descendants are geographically distributed allover a specific region such as a country. This hierarchy does not have fixed semantics which means that cloud providers can construct their hierarchies using a mix of domain and backend datacenters. Each datacenter is hierarchal in nature as master/slave configuration prevails. Hierarchical cloud organization reassures that domain and backend datacenters comply with the provider agreements, provide better mechanism for monitoring, error detection, fault tolerance and recovery. We detail benefits of the hierarchical organization for the cloud, we show that hierarchical organization provides the cloud with high availability and scalability rates and facilitates cloud management.**

***Index Terms*—Cloud computing, hierarchical organization, cloud tree, human collaboration, datacenters, network latency, response timem, times latencies, scalability, availability.**

I. Introduction

One of the major strengths of a hierarchical structure is that people are familiar with it. From universities to companies to government, people can find the hierarchical structure facilitates the workflow of their information. All information is hierarchy in one way or another, so storing or processing it within a cloud of a hierarchical structure complies with this nature. A cloud provider can achieve a better coverage ratio for his/her clients with low response time and minimal operational cost by organizing the cloud datacenters hierarchically. The hierarchical distribution of datacenters will help covering a larger geographic area such as a country or province thus reduces network latency.

The hierarchical distribution facilitates cooperation between datacenters for executing complex incoming requests thus reduces processing delay and provides high scalability. Also, it provides the cloud with a high reliability rate against hardware, software or power failures thus reducing times latencies and provides high availability rate. When using the hierarchical model for organizing cloud datacenters three questions may be raised:

- How to allocate resources, and manage them? In other words, how to ensure performance, reliability and availability.
- Scaling and time latency brings other key challenges. Tens of datacenters with tens of machines, failures are the default case!
- Load-balancing.

In the coming sections, we will answer in details those three questions. Fig. 1 shows a typical architecture of a datacenter which consists of a master server and *n* slave servers. The master machine server monitors the slave machines and acts as a load balancer.

Manuscript received May 13, 2013; revised July 18, 2013.

K. A. Nagaty was with the Faculty of Computer Science and Information Systems, Ain Shams University, Cairo, Egypt. He is now a Visiting Associate Professor with the Faculty of Computer Science at the British University in Egypt (e-mail: khaled.nagaty@bue.edu.eg).

II. Related Work

Time latency for real time services is considered an issue where the authors in [1], [2] showed that short time latency is required to maintain end-users satisfaction. The EdgeCloud technology presented in [3] is trying to find solutions for two main issues in the current cloud architecture which are the coverage ratio of clients and times latency. Using this technology the authors are augmenting the current cloud infrastructure with end-hosts or peers that are more geographically diverse than datacenters and have more specialized structures. However, it is difficult to find such peers or end-hosts which have the required specialized hardware or having the required software to serve the incoming requests, because some end-hosts may be busy or asking for high prices that may not be suitable for the cloud provider or for the client himself/herself, as one peer will be serving one client at a time.

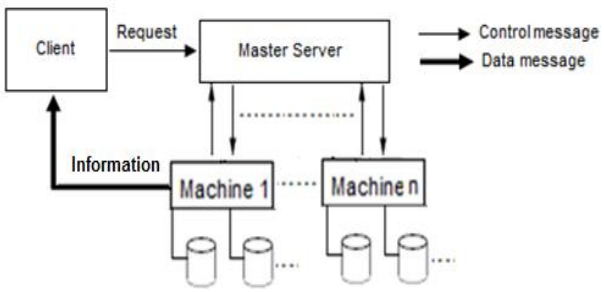

Fig. 1. Data center architecture

In [1], a peer is removed from the pool of peers if it is selected to serve a client this makes finding eligible peers more difficult especially if the number of incoming requests increased which makes the provider and the clients pay higher price for eligible peers. Some end-hosts or peers may not be geographically near the client which means that the

DOI: 10.7763/IJCEE.2014.V6.786

problem of time latency may still exist. Also, the reliability of the selected end-hosts or peers is considered an issue because serving a client from an unreliable peer can lead to a very poor user experience or leading to decisions with dramatic effects. One solution for the reliability issue of the peers is to let each peer to send a report to the coordinator at the end of each session and a reliability score is given to each peer reflecting the percentage of sessions that it successfully served [1]. However, by using this approach the reliability issue of the peers is not completely solved because they are not controlled by the coordinator and loosing peers because of reliability problems will limit the coverage ratio of the clients and increases time latency, this is in addition to the high price the cloud provider or the client should pay for eligible peers. The motivation of this paper is to solve the problems of clients' coverage ratio, times latency and reduce the cost of operation by modifying the current cloud organization to the hierarchical one. The hierarchical organization is shown to be better than both the flat or network organizations for information management which includes information creation, gathering, maintenance, flow, authenticity and security [4]. This paper is organized as follows: Section III presents the hierarchical organization of the cloud, Section IV presents the response time analysis of the cloud-tree, Section V presents the EC2 case, Section VI presents the benefits of using the cloud tree, Section VII is dedicated for conclusions and future work.

III. Hierarchical Cloud Organization

Real time applications running on the current cloud architecture suffer from times latencies. Also, clients across some geographic regions may not be covered by the cloud services. In order to solve these two problems we introduce a new cloud architecture which is the cloud tree. It is the hierarchical distribution of the datacenters over a geographic region such as a country. At the lowest level of the cloud hierarchy, there are the leaves, which are a pool of datacenters that covers a geographic region, we call them backend datacenters. At a higher level in the hierarchy there is a domain datacenter that controls the group of backend datacenters. In this section, we will describe the hierarchical cloud organization or the cloud tree.

A. Architecture

If the cloud infrastructure supports multiple datacenters across a geographic region then using the hierarchical architecture will help adding layers of redundancy, protection and load balancing. The hierarchical organization of the cloud is considered a latency-based strategy where the cloud provider places *k* datacenters at different hierarchical levels they are almost three, such that the number of covered clients across a specific region is the maximal [5]. In the proposed hierarchical cloud, regions with dense population are covered by a larger number of datacenters more than the regions with less population or they are covered by backend datacenters that have a larger computational power, which means larger number of servers with specialized hardware. A computational power of a backend datacenter is the average computational power of its servers, and the computational power of a server is its average throughput per second. The hierarchical organization of the domain and backend datacenters provides a better distribution of these datacenters to cover more clients and decreases the response time. The domain datacenters are distributed geographically and they combine resources from their backend datacenters to maximize cloud coverage and reduce processing delay. They are designed to be a shared platform for serving near real time applications. Clients close to a backend datacenter are served if it has the application required and fulfills the real time requirements. If a backend datacenter cannot serve the client, it sends a message to the domain datacenter to select a closer backend that has a copy of the application that meet the response time constraints. For example, in the cloud hierarchy of a country there is a master datacenter which can be placed in the capital of this country at level 0, a number of domain datacenters are placed in the capital of each province of the country at level 1. Each domain datacenter has a number of backend datacenters placed in the major cities of each province which is at the lowest level of the hierarchy level 2. The master datacenters placed in the capitals of different countries are geographically distant and fully connected as peer- to- peer. The benefit of using hierarchical datacenters, where a group of backend datacenters are controlled by a domain datacenter, is to protect your entire site/application from being negatively affected by some type of network/power failure, lack of available resources, or service out age that's specific to a particular datacenter. For example, in case of a power failure in one of the backend datacenters, the domain datacenter will reroute all the incoming requests to the failed backend to another working backend datacenter within its domain. Figure 2 shows an example of the proposed hierarchical organization of cloud computing in a country

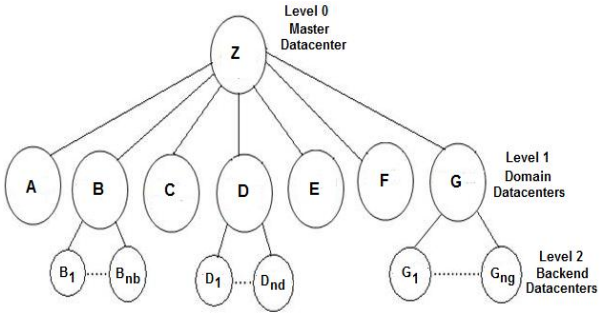

Fig. 2. A proposed hierarchical cloud organization for a country

where Z: is the capital, A-G: are provinces, B_1-B_{nb}, D_1-D_{nd} and G_1-G_{ng} are major cities within provinces B, D and G respectively with capital Z and seven provinces named A, B, C, D, E, F & G, where each province contains a number of major cities.

B. Hardwar Requirements

Cloud tree servers, located at a backend datacenter must have the sufficient computational power and the necessary specialized hardware such as GPU and fast memory to run applications. In addition, to hardware requirements, real-time software such as synchronous programming languages and real time operating systems are essential to run real time software applications. In addition to that, a sufficient

downstream bandwidth is required to receive the client's inputs and upstream bandwidth to send the application outputs to the client. The servers of the backend datacenters within a geographic region should support concurrent computing where the domain datacenter has a shared main memory or distributed memory. Each server of a backend can support multi-core computing where the output of each server is placed in the main memory of the backend datacenter. In case of concurrent processing the outputs from all backend datacenters are sent to the main memory of the domain datacenter. The architecture in Figure 3 shows a hierarchy of memory and communications between a domain datacenter and its backend datacenters with their servers. Each server contains many processors to support concurrent processing.

C. Example

Although there are many criteria to distribute the datacenters of a cloud among geographic regions hierarchically, in the example of figure 2 the provinces of a country are used to distribute the datacenters. The master datacenter of the cloud tree is placed in the capital Z which is at level zero of the cloud hierarchy. Each province of the country has a domain datacenter that is placed in its capital at level 1 of the tree. The master datacenter at the capital Z controls all the domain datacenters in all the provinces of the country, while each domain datacenter in a province controls many backend datacenters placed in the major cities of this province. The number of backend datacenters in a province depends on the population number, where the domain datacenter of a dense populated province should have a large number of backend datacenters to cover all the major cities within the province in order to minimize the times latencies beyond a specific threshold. When a client in a city within a province wants to access the cloud tree, it connects to the backend datacenter that is closer to his/her city. For example, a client living in city D_1 in province D can access the backend datacenter at D_1 to execute his/her request using the application running on this backend. If D_1 succeeded in executing the request it sends the output back to the client.

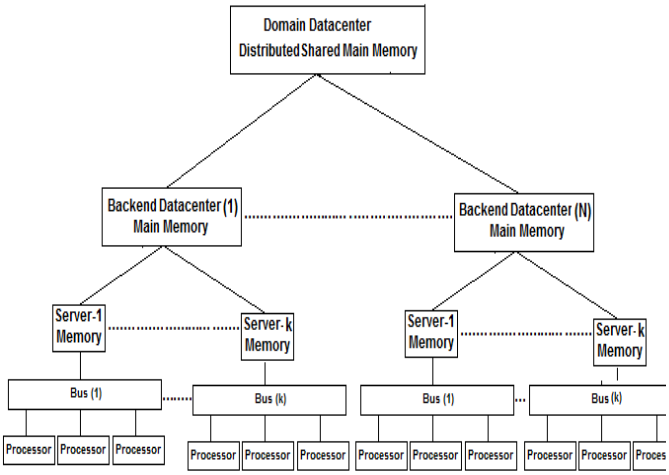

Fig. 3. Memory and communications hierarchy between a domain datacenter and its backend datacenters and their severs

If D_1 could not execute the request because more computational power or a specialized hardware or software is required, which care be found in another closer backend datacenters, then D_1 sends the request along with client's IP address to the domain datacenter in D. The domain datacenter in D searches for the eligible backend to execute the client's request. If distributed processing is required, the domain datacenter in D can distribute the tasks among several backend datacenters within its domain such as from D_2 to D_{nd}, where D_{nd} is the number of backend datacenters covering the major cities of province D. When all backend datacenters involved in performing the distributed tasks finished their jobs, they send back the output to the domain datacenter in the capital D, which in turn sends it directly to the client using the IP address sent previously by D_1. If the domain datacenter in D could not execute the client's request because a specialized hardware or software is required, it sends the request with the client's IP address to the master datacenter in the capital Z, which can involve another domain datacenter from the nearer geographic province which has the specialized hardware or software required. When the request is executed, the information is sent back to the capital Z which in turn sends it to the client using the IP address sent previously by domain D. In the worst case, if more computational power or specialized hardware or software still required and the master datacenter in the capital Z could not execute the client's request, then Z sends the request with the client's IP address to the master datacenter of the nearer geographic hierarchical cloud which may be in another country to get the more computational power or the hardware/software required. This shows that hierarchy cloud organization is highly scalable and flexible, which means that more computational power can be added or a specialized hardware/software can be easily found or borrowed when it is needed in a very limited time and the results are sent back to the client directly from the domain datacenter involved in the execution without the need to return back to the first backend datacenter that received the client's request.

D. Workload Distribution

Load balancing is the more interesting building block in cloud computing as it automatically distributes incoming client's requests across multiple cloud servers. Load balancing can be hardware or software. The most common use of load balancing is to provide cloud services to the clients either by using multiple servers of a backend datacenter or multiple backend datacenters of a domain datacenter. Each domain datacenter contains a load balancer to forward incoming requests to one of its backend datacenters. Also, each backend datacenter has a load balancer that forwards incoming requests to one of its servers. The servers or backend datacenters reply back to the load balancer which allows the backend or the domain datacenters to reply to the client without the client ever knowing about the internal separation of function. The hierarchical architecture of the servers within a cloud hides its internal organization and provides more security against attacks on the servers and prevents from running unrelated services. In case of all the servers of a backend datacenter are ineligible or unavailable because they do not have the

required hardware/software, the load balancer of the backend datacenter forwards the request to the load balancer of the domain datacenter which in turn forwards this request to a backend datacenter within its domain that has the available and eligible server. Another option for the load balancer to do is to display a message to the client regarding the outage. Load balancing provides the cloud tree with mechanisms to achieve a significantly higher fault tolerance. It can automatically allow the cloud tree to respond quickly to power failure, hardware/software failure and to any increase or decrease of clients' requests for the services running on its application servers. A variety of algorithms are used by load balancers to determine which server or backend datacenter to send a request to it. One of these algorithms is the round robin, while another approach is that servers with higher computational power and containing the required hardware and software are assigned a greater share of the workload than others. The pooling approach mentioned in [2] affects clients' coverage if only peers from one geographic region are selected where other clients from other parts of the country would be unable to find a nearby peer. In addition, this pool of candidates is of limited clients' coverage because one peer serves one client at a time, and in case of a large number of clients some clients may not find a peer. In the hierarchical organization of the cloud better client coverage is provided even if the client is not located geographically near a backend datacenter or a peer, the client's request with the IP address will be sent up the cloud hierarchy to the domain datacenter which will forward this information to another backend datacenter with eligible servers. The result information is sent by the domain datacenter or directly by the backend datacenter to the client in order to reduce network latency. In case of the domain datacenter could not find an eligible backend datacenter within its geographic domain, the client request and his IP address will be sent to the master datacenter to find an eligible backend datacenter within another domain. This makes all the backend datacenters of the cloud tree a pool of eligible candidates to execute the client's request.

E. Backend Selection

Domain datacenters use selection criteria to select the eligible backend datacenter to serve the incoming request. These parameters include:

- Latency: reducing time latency beyond the application's specified threshold is an essential target for the hierarchical cloud, therefore a domain datacenter must select a backend that is geographically closer to the client to reduce network latency and with suitable computational power to reduce processing delay.
- Application coverage: a domain datacenter preferentially selects the backend that stores the most popular applications and that respect the target application response time to the client.
- Cost of operation: a backend datacenter must select the appropriate server to execute the client's request, otherwise choosing a server with higher computational power than required may cause to postpone the processing of other incoming requests which need the computational power of this server or to reroute the request to the domain datacenter to find another eligible backend datacenter.

IV. Response Time Analysis

The response time $T_{response}$ can be formulated as follows:

$$T_{response} = T_{datacenter} + T_{server} + T_{client} \quad (1)$$

In this section, we provide a breakdown of the time latencies that form the total response time. We define $T_{\Delta\alpha\tau\alpha\chi\varepsilon\nu\tau\varepsilon\rho}$ as the delay to access the required backend datacenter that will execute the incoming request this may include the delay to access other datacenters in between the backend datacenter of the client's city and the eligible backend datacenter. The $T_{Datacenter}$ is formulated as follows:

$$T_{datacenter} = T^{F}_{backend} + T^{F}_{domain} + T^{F}_{master} \quad (2)$$

where:

$T^{F}_{backend}$: is the delay to access the backend datacenter of the client city from the client's home including network latency.

T^{F}_{domain} : is the delay to access the domain datacenter from the client's city backend datacenter including network latency.

T^{F}_{master} : is the delay to access the master datacenter from the domain datacenter including network latency.

In best cases T^{F}_{domain} and T^{F}_{master} are both equal to zeros, if the backend datacenter of the client's city has the server with the required computational power, both the hardware and software needed and thus the request is executed directly without accessing the domain datacenter. On the average case, the backend did not execute the request and the domain datacenter is accessed, in this case $T^{F}_{backend} > 0$ and $T^{F}_{domain} > 0$. In the worst case, the domain datacenter did not execute the request and the master datacenter is accessed and in this case:

$$T^{F}_{backend} > 0, T^{F}_{domain} > 0 \ \& \ T^{F}_{master} > 0$$

In other words:

Best case: $T_{datacenter} = T^{F}_{backend}$

Average case: $T_{datacenter} = T^{F}_{backend} + T^{F}_{domain}$

Worst case:

$T_{datacenter} = T^{F}_{backend} + T^{F}_{domain} + T^{F}_{master}$ We note that the time latency T^{F}_{domain} includes the time required by the domain datacenter to search for an eligible backend, T^{F}_{master} include the time needed by the master datacenter to search for an eligible domain.

The formula used to measure the network latency [3] is:

$$T(n) = \alpha + \frac{n}{\rho} \quad (3)$$

It is the time to send a message of *n* bytes in seconds where α is a constant and it is known as "zero bytes latency" and ρ is the network bandwidth. Network latency can be

reduced significantly when we consider the backend datacenters are closer to their clients in the cloud hierarchy. In addition, we define T_{server} as the processing delay, which refers to the time spent by the server or servers to execute the incoming request. The processing delay is affected by the amount of computational power provisioned by the backend datacenter. The work in [6] showed that this processing delay ranges from 10 ms to more than 30 ms and performance degrades when several virtual machines share the same machine. T_{server} is the time latency used to transmit the information back to the client, the same formula in [2] is applied to measure time latency in the network back to the client. The breakdown of T_{client} as follows:

$$T_{client} = T^{B}{}_{city} + T^{B}{}_{domain} + T^{B}{}_{master} \quad (4)$$

where:

$T^{B}{}_{master}$: is the delay to access the domain datacenter from the master datacenter.

$T^{B}{}_{domain}$: is the delay to access the client's city backend datacenter from the domain datacenter.

$T^{B}{}_{city}$: is the delay to access the client from the client's city backend datacenter.

V. EC2 Case

The Amazon EC2 cloud offers three datacenters in the US to its clients which makes more than one quarter of the population cannot play games from an EC2 cloud gaming platform with 80 ms network latency and covering ratio 70% , also almost one tenth of the clients cannot be reached at all [7]. The hierarchical cloud can solve these problems, where EC2 can distribute the computational power of its datacenters over a larger number of geographic regions each of which covers the clients of a specific region thus increasing the clients' coverage ratio. These geographically distributed datacenters will be closer to their clients more than the current datacenters thus reducing the network latency. Each group of datacenters within a geographic region is controlled by a domain datacenter which acts as load balancer to distribute the incoming requests among these datacenters thus reducing the processing delay T_{server}

VI. The Benefits of Using Cloud Tree

A. Effective Cloud Management

1) Load balancing which enables a domain datacenter to add more computational power from its backend datacenters if needed within minutes.
2) Cloud hierarchy organization reassures that backend datacenters will cooperate as follows:

Domain datacenters create and maintain oversight mechanisms to ensure the backend datacenters involve cooperation, competition, and communication between processes that either run simultaneously on their servers or are interleaved in arbitrary ways to achieve a near-real time response.

B. Facilitates Human Collaboration

The cloud hierarchy improves communications tools for people in that it facilitates collaboration which helps to save time, reduce duplication of work, and speed making decisions that could translate to more benefits.

C. Scalability

If more computational power is needed then the domain datacenter can reroute the task to a more eligible backend or distribute the task among several backend datacenters for concurrent execution. The results are returned back to the domain datacenter to be sent back to the client. Also, the provider can add more backend datacenters to the domain that experiences high incoming traffic or adding more servers to a backend which overloaded with jobs.

D. Availability

If a domain datacenter failed due to power failure or hardware/software failure for example then the master datacenter will redirect all the incoming traffic to the closer domain until the failed domain is recovered. In case of a failure backend the domain datacenter will redirect the incoming traffic to another backend until the failed backend is recovered. If a server within a backend is failed it is replaced by another server until the failed one is recovered. This substitution policy allows the cloud tree to achieve high availability of 24×7.

VII. Conclusion

In this paper, we presented a new organization for cloud computing that is based on the hierarchical model. Using the hierarchical organization the datacenters of the cloud are better distributed among several geographic regions which provide a better coverage ratio of their clients. The substitution policy adopted by the cloud tree provides high availability of the cloud. The selection criteria applied by the cloud tree to find an eligible datacenter to reroute the client's request to it, provides high scalability where more computational power is added when needed. To reduce the network latency, the backend or domain datacenters which executed the incoming request will respond directly to the client. In our future work, we will explore the effectiveness of using the hierarchical paradigm for cloud datacenters and compare it by other new paradigms such the EdgeCloud.

References

[1] K. Chen, P. Huang, G. Wang, C. Huang, and C. Lei, "On the Sensitivity of Online Game Playing Time to Network QoS," in *IEEE INFOCOM*, 2006.

[2] M. Claypool and K. Claypool, *Latency Can Kill: Precision and Deadline in Online Games*, pp. 215–222, 2010.

[3] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "EdgeCloud: A New Hybrid Platform for On-Demand Gaming," Technical report CS-2012-19, University of Wateloo, Canada, 2012.

[4] K. A. Nagaty, "Hierarchical Organization as a Facilitator of Information Management in Human Collaboration," *IGI Global*, ch. IV, 2009.

[5] M. Claypool and K. T. Claypool, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, 2006.

[6] R. Hockney, "The communication challenge for MPP: Intel Paragon and Meiko CS-2," *Parallel Computing*, vol. 20, no. 3, pp. 389-398, March 1994.

[7] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. "An Evaluation of QoE in Cloud Gaming Based on Subjective Tests," presented in International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2011.

Khaled A. Nagaty was born in 1960 in Cairo, Egypt. He got his B.SC. in Statistics from Cairo University in 1982, then he got his Master degree in computer Science from the same University in 1990. He got his Ph.D. in Computer Science in 1999 from Cairo University Egypt. He is an Associate Professor of Computer Science at Ain Shams University but currently he is a Visiting associate professor at the British University in Egypt. His research interest include pattern analysis, image processing, cloud computing and computer security.