What web objects contained in popular websites are delivered through hypergiants.

Parida,Soumya Ranjan111

October 2, 2016

Abstract

With the proliferation of the Internet,hypergiants such as Google ;Content delievery networks like Akamai etc. often play a vital role in providing the content of any website. These hypergiants not only provide different services but also rich in content. Flash medias from Youtube, login system from Google,Facebook advertisements from Google ad sense,popular social media sites like Facebook,Twitter,LinkedIn etc. are common and popular services embedded in most of the websites today.

To cope with this demand, hypergiants have been deploying a large number of scalable and cost effective hosting and content delivery infrastructures all around the globe. These hosting infrastructures can be composed of a few large data centers ,a large number of caches or any combination. Such a scenario cause a large amount of traffic flow from hypergiants as well as huge dependency between popular websites and hypergiants.

In order to know how the involvement of popular websites with hypergiants is evolving,this thesis addresses the following research questions .Firstly,are there any presence of these hypergiants in interenet architecture ? If there are hypergiants present,then what percentage of popular web sites out of 100,000 top websites of alexa are connected with various hypergiants ? Thirdly ,What percentage of different objects (text/html ,img ,script,media etc.) are connected with hypergiants ?

This thesis provides a quantitative research of web connectivity for alexas top 100,000 websites. We present the design, implementation and analyses of different types of objects contained in various websites which are connected to different hypergiants . The thesis follows two paths. Firstly ,in order to quantify different types of objects involved in websites ,we will crawl home pages of top 100,000 websites of alexas website and gather the presence of hypergiants infrastructure linked to different objects like images,external links,scripts ,embedded videos,css files etc. in those websites. Secondly, We examine those objects to find out the degree of connection between the top 100,000 websites and hypergiant.

The experimental results discussed in this thesis are supported by extensive analyses of data collected which provide evidence in support of the conclusion provided in this thesis.

Abstract

With the proliferation of the Internet,hypergiants such as Google ;Content delievery networks like Akamai etc. often play a vital role in providing the content of any website. These hypergiants not only provide different services but also rich in content. Flash medias from Youtube, login system from Google,Facebook advertisements from Google ad sense,popular social media sites like Facebook,Twitter,LinkedIn etc. are common and popular services embedded in most of the websites today.

To cope with this demand, hypergiants have been deploying a large number of scalable and cost effective hosting and content delivery infrastructures all around the globe. These hosting infrastructures can be composed of a few large data centers ,a large number of caches or any combination. Such a scenario cause a large amount of traffic flow from hypergiants as well as huge dependency between popular websites and hypergiants.

In order to know how the involvement of popular websites with hypergiants is evolving,this thesis addresses the following research questions .Firstly,are there any presence of these hypergiants in interenet architecture ? If there are hypergiants present,then what percentage of popular web sites out of 100,000 top websites of alexa are connected with various hypergiants ? Thirdly ,What percentage of different objects (text/html ,img ,script,media etc.) are connected with hypergiants ?

This thesis provides a quantitative research of web connectivity for alexas top 100,000 websites. We present the design, implementation and analyses of different types of objects contained in various websites which are connected to different hypergiants . The thesis follows two paths. Firstly ,in order to quantify different types of objects involved in websites ,we will crawl home pages of top 100,000 websites of alexas website and gather the presence of hypergiants infrastructure linked to different objects like images,external links,scripts ,embedded videos,css files etc. in those websites. Secondly, We examine those objects to find out the degree of connection between the top 100,000 websites and hypergiant.

The experimental results discussed in this thesis are supported by extensive analyses of data collected which provide evidence in support of the conclusion provided in this thesis.

Contents

1 Introduction

The Internet has changed a lot within last decade both in technical as well as in user experience aspects.With increase in internet users and their demand for more and richer content has led to exponential increase of internet traffic.Social networking sites like facebook,twitter enable users to publish their own content and share with other users.Users also share videos in different social media sites like youtube,facebook etc.The highly popular on-demand video and streaming sites like netflix etc., are also playing vital role in increasing internet user base and traffic.Recent traffic studies [3,4] show that a large fraction of Internet traffic is originated by a small number of prominent infrastructure which can be highly distributed CDNs like akamai or content providers like google.Poese et al. [3] report a similar observation from the traffic of a European Tier-1 carrier. Labovitz et al. [2] infer that more than 10 % of the total Internet inter-domain traffic originates from Google, and Akamai claims to deliver more than 20 % of the total Web traffic in the Internet [5].

Consequently,popular websites require different strategies to distribute their content all over the world while offering the best customer service.Leighton [1] proposes four main approaches to distributing content in a content-delivery architecture: (i) centralized hosting,(ii) big data center based CDNs (content-delivery networks), (iii) highly distributed CDNs, and (iv) peer-to-peer networks.In cerntarlized hosting case ,traditional architectured sites take help of one or small number of collocation sites to host their content.These centalized hosting structures are may be enough for small content sites but not for popular websites which carries hugh amount of content.Big Data Center content delievery networks have hugh number of high-capacity data centers which are connected to major backbones.Highly popular content are cached.Hence able to increase the performance of delievery compare to centralized hosting infrastructures but still are limited in potential improvements as still they are far away from the end user.Third type of model is highly distributed content delievery networks.They have high footprint all over the world.By putting their own infrastructures inside end user's ISP,they are able to eliminating peering, connectivity, routing, and distance problems, and reducing the number of Internet components.Final approach is peer to peer networks which has very little scope in delievering the content of popular websites in today's internet world due to serious concern of the copy right issues.

Traditional hosting model like content delivery networks (CDNs) are the most important technical solutions for providing high performance delivery system till now where popular contents are stored in servers of CDNs .But with the increase of content within sites ,it is not possible for the popular web sites to provide better performance to end customers by using only the traditional hosting model.Instead ,content providers now build their own global backbones, cable Internet service providers offer wholesale national transit, and transit ISPs offer CDN and cloud / content hosting services.CDNs also build highly distributed infreastructures and data centers to replicate the most popular content at different distributed cache servers and locate them at the edge of the network.It help them to provide popular content from the nearest server to customers.Hence when a user request for a popular web content ,CDN just redirect the user to most suitable server by bypassing the saturated links.

Hence due to this change in content delivering phenomenon some researchers termed this companies as Hyper Giants [6] which include large content providers, such as Google and Yahoo, as well as highly distributed content distribution networks (CDNs), like Akamai,big cloud computing CDNs like Amazon aws etc. Most of these hypergiants are operating not only a substantial number of data centers but are also building up their own network [21]. Some networking researchers are claiming that, due to the phenomenal growth of hypergiants, the topological structure of the internet must be redrawn to include them, together with the Global transit and backbone networks as part of the Internet core,resulting in the topology sketched in Figure 2. This may leave the ISPs as dump pipe providers to the consumer.

Again it is important to understand that hypergiants are not usually the main operators of the network but they play vital role in delievery of the content by creating interdependecy between them and the main operators by different ways and business needs. The producers of the content (popular websites) want their content to be delievered to end user in less time for which they have to rely on the main operators or hypergiants. Such a scenario cause a large amount of traffic flow from hypergiants as well as huge dependency between popular websites and hypergiants.It is this symbiosis between the two parties that motivates our work ,giving an overview on how far the reach of hypergiants in todays internet.

Again hypergiants now not only provide rich content,they also provide differnt other services.For example now most of the popular website having their own login systems or login systems which are provide through google,twitter,facebook etc.In later case the authentication is verified by google,facebook etc.For that the popular websites needs to embed third party login systems in their websites.Like this there are lot of other services provided by these hypergiants like for advertisements websites add google adsense etc.Again popular websites store different content type like image files ,video files ,audio files in highly distributed CDNs.This dependency of popular websites on hypergiants also gives us motivation to check what percentage of these hypergiant objects are linked with popular websites.

A few studies have already investigated about hypergiants and their relationship with popular websites in the recent past.In 2010, Craig Labovitz, then of Arbor Networks [2],defined a new type of network entity. By placing Google in this list,he characterized the hypergiant as a content provider that makes massive investments in bandwidth, storage, and computing capacity to maximize efficiencies and performance.The concept of hypergiants also aligns with Schmidts assertion which talks about "gang of four" companies which are responsible for the growth and innovation of internet. Google, Apple, Amazon, and Facebook[7].Bernhard et al.[11] also worked on identifying and mapping the content infrastructures that are hosting the most popular content.Along with this author purposed a light weight automated technique to find out the hyper-

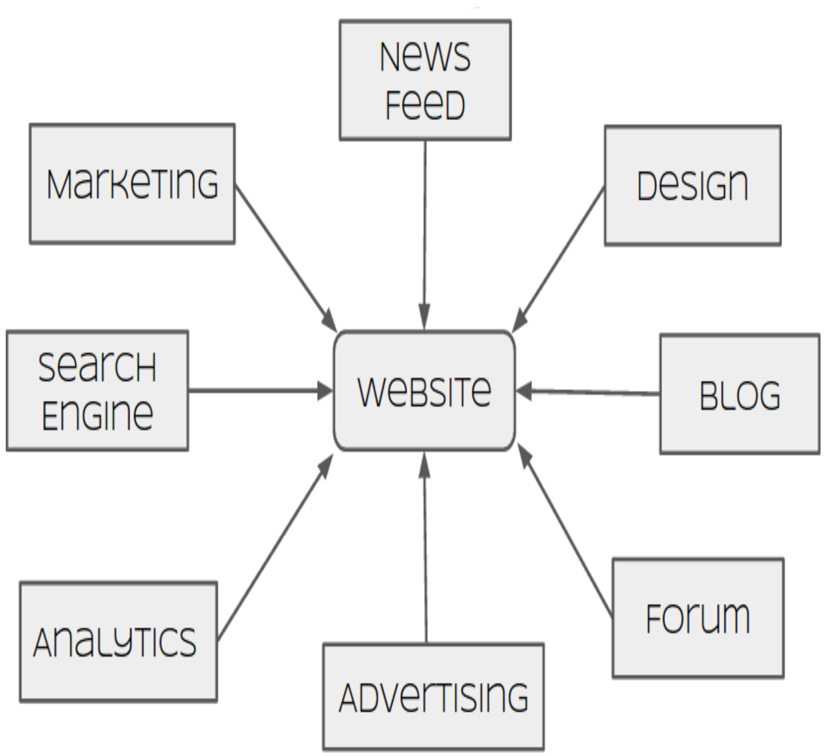

Figure 1: website dependency on hypergiants

giants and other high distributed CDNs based on their footprint.Again author considered high distribued CDNs,distribued CDNs different from hypergiants where he identified hypergiants based on their prefixes-ASN mapping.Gao [4] have also analyzed operator interconnections from a more technical perspective. They use a methodology to quantify the type of inter-Autonomous System (AS) relationships that exist in the Internet and classify them into three groups based on the state of Border Gateway Patrol (BGP) messages: customer-to-provider, peer-to-peer, and sibling-to-sibling relationships.Shavitt and Weinsberg recently discussed the topological trends of content providers. They create a snapshot of the AS-level graph from late 2006 until early 2011, and then analyze the interconnection trends of the transit and content providers and their implications for the Internet ecosystem. AS graphs are built by traversing IP traceroutes 9 and resolving each IP address to its corresponding AS.Shavitt and Weinsberg proved that large content providers like Google, Yahoo!, Microsoft,Facebook, and Amazon have increased their connectivity degree during the observed period and are becoming key players in the Internet ecosystem, strengthening the idea that the Internet is becoming flatter.Palacin et [13] defined hypergiants not only content providers they are basically content aggregators. This means that these hyper-giants are attracting content from smaller sites or individuals and publishing it via their high-speedinfrastructures. Somehow a cannibalization process has begun in which the hyper-giants are absorbing content from the long tail, entering fully into the niche of the traditional hosting companies but the author mainly focused on the realtionship between content providers and ISPs.

By end of this thesis we are able to give answers for some of the important research questions which can be summarized as follows:

- Identification of hosting infrastructures: We propose a lightweight and fully automated approach to discover hypergiants such as highly distributed content delievery networks,content providers etc. based on DNS measurements and BGP routing table snapshots.
- Classification of hosting infrastructures: We classify indi- vidual hosting infrastructures and their different deployment strategies based on their network.
- Web content dependency: We quantify the degree of content dependecy of the popular websites on hypergiants.

This remainder of this thesis is structured as follows.This thesis is separated into 7 chapters.

- Chapter 2 :It starts with the evolution of internet architeure from early 2000s to current time and how the dependency of popular websites on hypergiants increases with time.
- Chapter 3 :This section will provide the overall methodology used in this thesis.
- Chapter 4 :This section focuses on the details about environment and technologies used for the prototype. The implementation details of web crawler engine, its operations and configuration management are explained.
- Chapter 5 describes the measure details.
- Chapter 6 describes the problems encountered during the thesis work, learnings during this phase, any solutions to overcome the problems encountered. It also summarizes the results and includes possible future work.

2 Background

In this chapter,we discuss how internet architecture evolved with time.Along with this we will dicuss briefly on hypergiants and different technical researches already done on hypergiants.In addition to this we also provide the techical background of dns and http protocol which will be used in this thesis.

2.1 Evolution of Internet Architecture and Rise of hypergiants

In 2010, Craig Labovitz, then of Arbor Networks [2],defined a new type of network entity.He argued transcended traditional content versus carrier dichotomy of internet architecture.By placing Google in this list,he characterized the hypergiant as a content provider that makes massive investments in bandwidth, storage, and computing capacity to maximize efficiencies and performance.The concept of hypergiants also aligns with Schmidts assertion which talks about ”gang of four” companies which are responsible for the growth and innovation of internet. Google, Apple, Amazon, and Facebook[7].

With the proliferation of the Internet, the internet architecture also evolved with time.The Internet architecture implemented until the early 2000s was based on a multi-tier hierarchic structure shown in figure-2.Tier 1 ISPs were on top of the hierarchy followed by the Tier 2 regional ISPs and the Access ISPs at the lower part of the hierarchy connecting the end users. In this scheme, Tier 1 ISPs were highly connected to other ISPs and offered transit services to other ISPs in lower layers.Content was distributed through Access ISPs or, in the best cases, through ISPs located at advantageous points. Traffic flows were required to go up and then down in the hierarchy to reach end users.Among the different network operators,internet traffic was exchanged at differnt IP exchange points according to agreements between different layer players where the dissymmetry in traffic was compensated.

But with the time,the internet architecture changed.Reaserchers found that now nobody has control over internet ,instead each ISP has control over its network and depend upon the network connected with it.Even during last decade the old pyramidal structure of internet architecture show in figure-2 has been bypassed by big content providers, such as Google, Facebook, Amazon or Yahoo!, and content delivery network operators, such as Akamai.As a result now internet's backbone has a flatter structure where there are few autonomous systems are playing major role in deliverying content.They are connected to each other and have a bigger footprint by establishing small data centers all over the world.This help them to get as close as possible to the access networks used by their customers,bypassing intermediate Internet service providers which can be seen from figure-3.The trend towards flatter network architectures can also be found in the area of access networks.The researchers termed these infrastructure providers as Hyper Giants which include large content providers, such as Google and Yahoo, as well as highly distributed content distribution networks (CDNs), like Akamai [6].

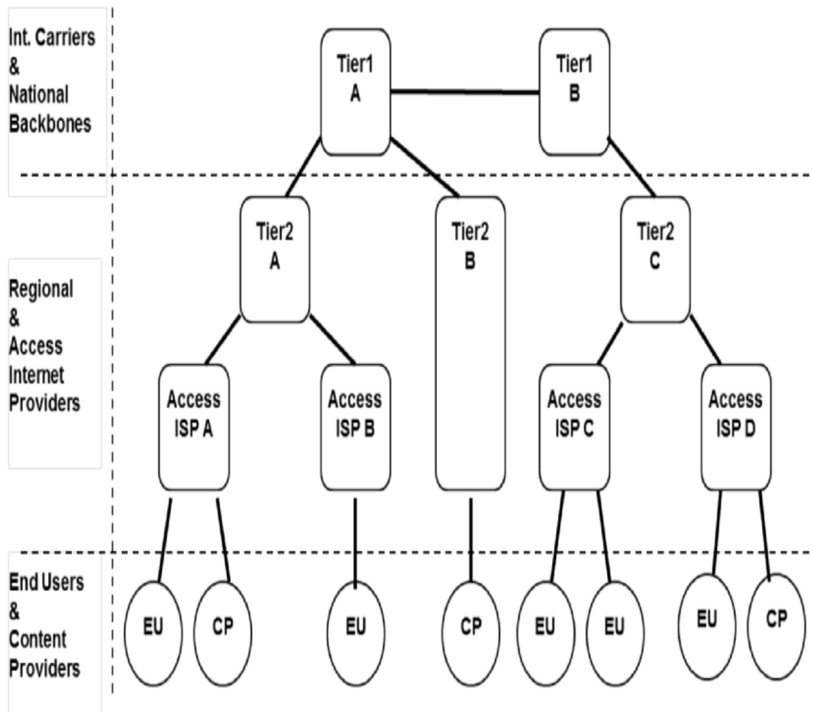

Figure 2: Traditional Hierarchic Internet Structure

The study on hypergiants can be found from "Web Content Cartography" paper where the author identified and classified the content hosting and delivery infrastructures including hypergiants present in internet.Along with this the author tried to quantify the degree of content replication in the Internet and its impact on local content availability in different regions of the world [8].

2.2 Protocols

In this section we will discuss about the protocols used in this thesis which are domain name system (DNS) and hyper text transfer protocol (HTTP).Both protocols used in out thesis extensively which will be discussed briefly in chapter 5.2.

2.2.1 Domain name System (DNS)

Domain name system (DNS) is used to translate IP address to corresponding host names.Internally it is maintaining a hierarcial structure of domains.Before the invention of DNS on year 1983,a simple text file (hosts.txt) file was used to do this translation from IP address to host name.But with a growing number of

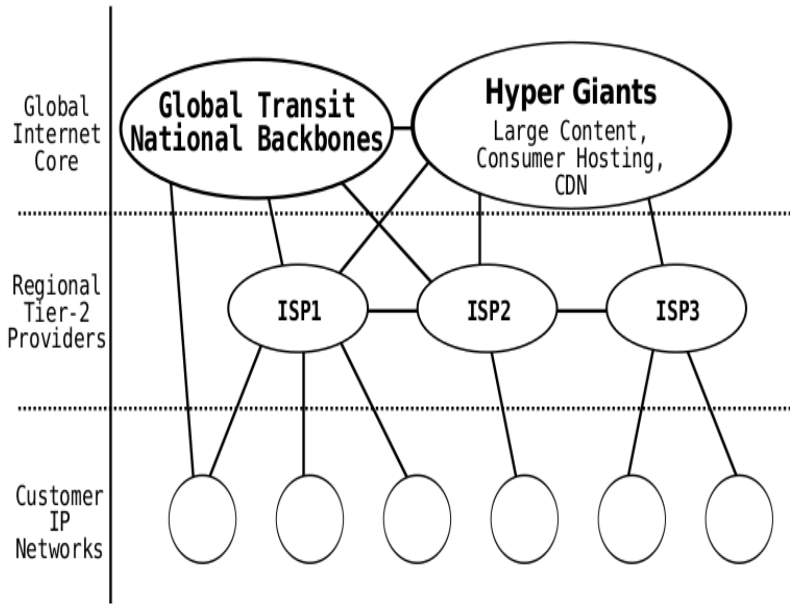

Figure 3: Modern Internet Structure

host names it was difficult to keep maintain in hosts.txt file and Domain name system introduced.

The administration of domains is divided into different zones. The zone information is distributed using authoritative name servers.The top most level of DNS starts with root zone and the root zone information is served by root servers.Responsibity of specific parts of zone can be given to some other authoritative name servers which in turn divided reponsibity with other authoritative name server.For, e.g.,the responsibity of .org domain is delegates to the Public Interest Registry by the root zone which in turn delegates responsibility for acm.org to the Association for Computing Machinery (ACM).At the end its site is responsible for its own zone and keep maintain its own database of authoritative name server.The information about a particular domain of a zone is kept in Resource Records (RRs) which specify the class and type of the record as well as the data describing it.Multiple RRs with the same name, type and class are called a resource record set (RRset).

To resolve a IP address into host name,the procedure starts with the end user's stub resolver queries to local name server called caching server.if caching server can't able to resolve it,it redirects the query to authoritative name server of the domain.If resolver doesn't know how to contact the corresponding authoritative name server of the domain,it redirects the query to root name server .The root name server again refers the resolver to the authoritative name server responsible for the domain just below the root server.This procedure continues till resolver is able to resolve the domain properly.

```
; <<>> DiG 9.9.5-3ubuntu0.8-Ubuntu <<>> www.bmw.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65119
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.bmw.com.                   IN      A

;; ANSWER SECTION:
www.bmw.com.            3600    IN      CNAME   cn-www.bmw.com.edgesuite.net.
cn-www.bmw.com.edgesuite.net. 3600 IN   CNAME   a1586.b.akamai.net.
a1586.b.akamai.net.     20      IN      A       104.121.76.64
a1586.b.akamai.net.     20      IN      A       104.121.76.73

;; Query time: 22 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Fri Sep 30 18:13:28 CEST 2016
;; MSG SIZE  rcvd: 143
```

Figure 4: DNS reply for a host using dig command line tool.

Figure-4 shows the DNS reply by the resolver when querying a host name served by a content infrastructure.Here the host name is www.bmw.com.The answer section contain a chain of CNAMEs which resolve into two ARecord set (RRset) with different IP addresses which can be for used for load balancing.

2.2.2 Hyper text transfer protocol(Http)

Hyper text tranfer protocol (HTTP) is an application layer protocol mainly used as defacto standard to transport content in world wide web.HTTP works on top of the TCP/IP protocol and follows the client server architecture via request-respnse communication procedure.It allows end-users to request, modify, add or delete resources identified by Uniform Resource Identifiers (URIs).

HTTP message consists of http header which shows the meaning of message

```
HTTP Request

GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 [...]
Accept: text/html,application/javascript [...]
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP Response

HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Type: text/html; charset=UTF-8
Date: MON, 12 Sept 2016 03:12:03 GMT
Last-Modified: MON, 18 Jan 2016 11:25:43 GMT
X-Cache: HIT
Content-Length: 11690
<!doctype html>
<html>
[...]
```

Figure 5: HTTP request (top) and response (down) for www.example.com

and http body which is actual message.HTTP message can be a request message or response message.The http client sends a request message to server .There are different types of methods used in http request message like GET,HEAD,POST ,PUT,DELETE,CONNECT etc.But in this thesis we have used extensible GET method and the HEAD method.The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.Same as GET, but it transfers the status line and the header section only.The introductory line in an HTTP request (Figure 5) consists of a method, a server-side path, and the HTTP version in use.The introductory line in an HTTP response (Figure 5) starts out with the HTTP version in use, followed by a standardized three-digit status code and a textual status description. The status code tells the requester about the success of the query or indicates the reason of an error.

Both HttP request and response starts with introductory header line which can be seen in figure-5.For the request ,it consists of a method and the URI it should act upon. Similarly, the introductory line of a response, contains a standardized three-digit status code and a response status code which shows whether the request was successful or not.Both request and response messages are followed by multiple header lines.Some header information are valid for request ,some are for response and some are valid in both the ways.Since HTTP1.1,the Host header is mandatory for request messages.The meta information encompasses information about the file type, the character set in use, preferred language etc.HTTP also allows server to set cookies in clinet side which help the server to track client requests.

2.3 Conclusion

Within last decade the internet architecture changed vastly due to the introduction of hypergiants which can be highly distributed CDNs,cloud computing CDNs etc.To understand the hyperginats,we need to see their role in internet traffic.Today's internet traffic is dominated by HTTP traffic.Again to deliever the content fast ,DNS protocol is used as the load balancing mechanism by these big hypergiants.

3 Methodology

In this section we will discuss about the whole approach to identify the hypergiants presence in today's internet.The key idea is to collect IP addresses that DNS retruns for alexa top 100,000 wevsites and the website links embeded in these top 100,000 websites.We will use the content type of all the http request and find out the different object type (text/html,image,video,audio etc) delivered by these hypergiants for the popular webistes and the embeded links in those websites.

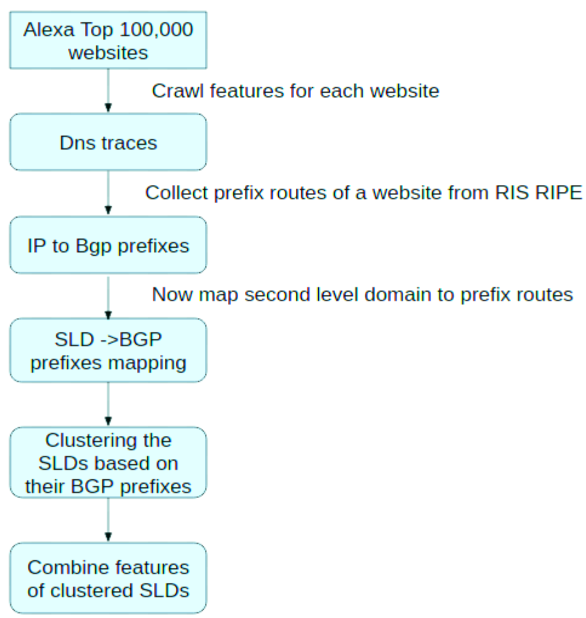

Figure 6: High level approach design approach

To acheive our goals of identifying and classifying hypergiants in internet,we divided the whole process into 6 parts shown in figure 6.We now elaborate the steps we followed ,choices we have taken to achieve our goals.

3.1 Hosting Infrastructure Coverage

To acheive our goals to find out the hypergiants in internet,we will use the hosting infrastructures which are serving popular websites.Given that internet traffic normally consistent with the Zipf's law [21],it is very likely possible that the hypergiants are the very popular content delievery networks or content providers etc. which is responsible for the mojority of the HTTP traffic flow in today's internet.For example, Akamai claims to deliver about 20% of the total Web traffic in the Internet [5]. According to Labovitz et al. [2] ,google serves up to 10 % of all Internet traffic.Similarly top 10 hosting infrastructures servees more than 15% and top 100 responsible for more than 40% of the traffic.In this thesis we will take the alexa top 100,000 popular websites.

3.2 DNS traces

We base our study on dns traces collected within single vantage point within europe.so this thesis will provide a better overview of hypergiants behaviour within our vantage point location in europe.As hypergiants are normally vast hosting infrastructures,content providers etc ,their footprint mostly present all over the world .Hence the identification of hyperginats should not be affected much due to this limitation.But there are few other research questions arises due to this network coverage limitation .These questions are mainly due to single vantage point instead of multiple vatage point.We will discuss about these research question in "Future Work" chapter.For collecting data we use "scrapy" framework.Scrapy framework is an open source web crawler.This is shown by second stage in figure 6 where top 100,000 popular websites of alexa are passing to the second stage where scrapy framework crawl the main page and the embeded pages linked inside the main page of each popular website of alexa.

3.3 IP to BGP Prefixes Mapping

We extract IP addresses of each website link using scrapy crawler.Along with this we also collect the ARecord names associated with the corresponding IP address.From ARecord names we collect the SLDs.The set of IP addresses for a perticular SLD shows the degree to which the corresponding SLD infrastructure is network-wise and geographically distributed.Hence natural choice for the features to consider are IP addresses,AS Numbers and the bgp prefixes.The number of bgp prefixes shows the network footprint and the number of ASNs shows how infrastructures are distributed all over world.For example the highly distributed infrastructures have a lot of prefixes as well as high number of ASNs.Similarly small datacenters will be located within a single AS in a single geographic location, having a limited number of bgp prefixes,and a large number of IP addresses.Eventually these features are coreleated and using these features we will try to find out the hypergiants presence in internet.

To determine bgp prefixes we use BGP routing information from RIPE RIS [22].The bgp prefix information for IP address 104.121.76.73 is shown in the figure 7.As we can see from figure-7, the information providered are like routes which is the bgp prefixes,origin shows the correcponding AS number.From the figure this can be seen that the ip address 104.121.76.73 can be routed via two different prfixes 104.64.0.0/10 and 104.121.76.0/24.So we will consider both the prefixes for the IP address 104.121.76.73.

```
whois -h riswhois.ripe.net 104.121.76.73

% This is RIPE NCC's Routing Information Service
% whois gateway to collected BGP Routing Tables, version2.0
% IPv4 or IPv6 address to origin prefix match
%
% For more information visit http://www.ripe.net/ris/riswhois.html
%
% Connected to backend ris-whois04.ripe.net

route:        104.64.0.0/10
origin:       AS35994
descr:        AKAMAI-AS - Akamai Technologies, Inc., US
lastupd-frst: 2016-07-07 14:54Z  198.32.160.39@rrc11
lastupd-last: 2016-10-02 06:12Z  193.203.0.130@rrc05
seen-at:      rrc00,rrc01,rrc03,rrc04,rrc05,rrc06,rrc07,rrc10,rrc11,rrc12,rrc13,rrc14,rrc15,rrc16,rrc18,rrc19,rrc20,rrc21
num-rispeers: 162
source:       RISWHOIS

route:        104.121.76.0/24
origin:       AS20940
descr:        AKAMAI-ASN1 , US
lastupd-frst: 2016-07-06 10:09Z  198.32.124.146@rrc16
lastupd-last: 2016-10-02 01:01Z  197.157.79.173@rrc19
seen-at:      rrc00,rrc01,rrc03,rrc04,rrc05,rrc06,rrc07,rrc10,rrc11,rrc12,rrc13,rrc14,rrc15,rrc16,rrc18,rrc19,rrc20,rrc21
num-rispeers: 164
source:       RISWHOIS
```

Figure 7: RIPE RIS bgp prefixes usinh whois command

3.4 SLD to BGP Mapping

From the second stage show in figure-6,we receive IP address and corresponding ARecords.From the ARecord,we get second level domain which gives us idea about the infrastructure invoved with the website.Again from stage-3 ,we get the IP address to BGP prefixes mapping using RIPE RIS file.In this stage we will create the mapping of SLD infrastructures collected in stage -2 to BGP prefixes collected in stage-3.Now we have all features IP addresses,BGP prefixes,ASN number which are required for detemine the type of infrastructure.

3.5 Clustering Algorithm

The clustering algorithm can be divide into two parts.In the first step ,we will try to aggregate the prefixes of a SLD into the set of parent prefixes and in the subsequent step we will compare prefixes of the SLDs with each other and cluster them if they are sharing most of the infrastructure.

3.5.1 Aggregate Prefixes

From the above steps,we collected SLDs with corresponding number of links served by SLD,number of IP addresses,bgp prefixes and number of ASNs.Now first we will sort the SLDs according to their number of links in decreasing order.This will give us all the popular SLDs which served maximum number of links.Now from the bgp prefixes,keep only parent prefixes and the prefixes which dont have any child in the prefix set.for example, googledomains.com has the prefix set ('216.239.32.0/19', '216.239.32.0/24', '216.239.34.0/24', '216.239.36.0/24'), '216.239.38.0/24').After the above step the prefix set will become ('216.239.32.0/19']) as other prefixes are subset of parent prefix 216.239.32.0/19.This procedure will be done for all the SLDs staring from first SLD sorted by decreasing order of number of links.Now starting from first and compare each SLD with the other SLDs.Between two prefix sets if child prefixes present then replace with parent prefix.This will make two prefix set with only parents.Now compare the two sets of prefixes to find out whether they are sharing same infrstructure or not.More details about the similarity procedure in next section.

3.5.2 Similarity between two prefixes set

Based on the similarity between two prefix set,we will decide if they belong to same SLD infrastructure or not.For this we define the similarity between two prefix set as follows,

$$similarity(s1, s2) = \frac{|s1 \cap s2|}{|s2|} \tag{1}$$

where s1,s2 are the bgp prefix sets.

If the similarity between two prefix set are greater than equal to 70% then we cluster both the SLDs together.Here we assume than if s2's 70% prefixes are presen in the common infrastructure between s1 and s2 set,then it shows that s2 sharing most of the infrastructure of s1.Hence we club them together.We will continue this procedure for all the SLDs string from the first SLD sorted by number of links. If two SLDs are matched them it will be removed from somparison with other SLDs and it will mapped to the SLD with which it matched.For example 'googleusercontent.com' matched with 'google.com' with similarity 100%.Once this is matched we clubbed googleusercontent.com with google.com and won't be available for any further similairty with other SLDs.70% of similarity is choosen after extensively testing between bgp prefixes.But this can be taken for future work to find out whether this number is correct one or because of this we miss some SLDs.

3.6 Combine features of clubbed SLDs

Now after the cluster algorithms, we will get prominent SLD infrastructures under which multiple number of SLD mapped to based on simiarity comparison between the bgp prefixes they routed to.Hence now we will club the features of the child SLDs with the parent SLDs.After this stage we have all the prominet infrastructures with number of unique bgp prefixes by both parent SLD and child SLDs,unique links by both parent SLD links and child SLDs ,unique ASNs both parent SLD links and child SLDs.

3.7 Conclusion

In this section we discussed about the whole method we are going to follow to reach our goals.Along with this we identify some of the future works which will be discussed breifly in "Future work" section.

4 Implementation

This section describes how the first prototype of an crawler was built. It explains the choice of programming language as well as go into some details of the objects, the rough internal working and their interaction with each other. In the end, the usage of the program will be explained briefly.

4.1 Choice of programming language

Before going for implementation,it is imporatnt to know that the whole process invove two parts.First one is crawling of top websites of alexa and second part is make an data analysis tool.Hence we need to choose a programming language which should be powerful in development as well as should be free, open and usable by anyone who wishes to run this crawler implementation.Again the language should have many open libararies .For this purpose we choose python as it fulfilsall necessary requirement.

4.2 Development Tools

For development,it is important to choose correct IDE which allows us to write code in less time with minimum effort.Along with this it helps in code completion, syntax highlighting, debugging and refactoring.For this purpose we have chosen "sublime" editor which is free and easy to write python code.

Now second most important aspect is to choose python web crawling tool which will suit our requirement and can be used in future.We will focus on programs that request web services from service providers and programs that scrape data from web sites. Web service applications will involve us in a new kind of programming called client-server programming; the programs we will look at will be client programs making requests from service on the Internet. Although the underlying foundation of a web-scraping program is also a client-server interaction, we will use some tools that hide the details of those interactions, and allow us to fetch web page content directly.For this we have couple of choices which are well known web crawling tools available like urlib,beautiful soup,scrapy etc. But Python Scrapy is the best out there, Scrapy crawling is faster than any other platforms, since it uses asynchronous operations (on top of Twisted). Scrapy has better and faster support for parsing (x)html on top of libxml2. Scrapy is a mature framework with full unicode, redirection handling, gzipped responses, odd encodings, integrated http cache etc.Again it is open source having a big community and documentation.

Lastly For data analysis we use the Spark Python API (PySpark).There are lot of different machine learning tools available but as we expect out data to be of some gigabytes,it is better to use that tool which can process data faster.Python and R are popular languages for data scientists due to the large number of modules or packages that are readily available to help them solve their data problems. But traditional uses of these tools are often limiting, as they process data on a single machine where the movement of data becomes time consuming, the analysis requires sampling (which often does not accurately represent the data), and moving from development to production environments requires extensive re-engineering. To help address these problems, Spark provides data engineers and data scientists with a powerful, unified engine that is both fast (100x faster than Hadoop for large-scale data processing) and easy to use. This allows data practitioners to solve their machine learning problems (as well as graph computation, streaming, and real-time interactive query processing) interactively and at much greater scale.

Apart from above tools we use some other tools for the implemenatation.The following are the list of important softwares and tools used.

- dnspython:it is a DNS toolkit for python.This is used in the code to get the A records of hosts.
- pygeoip :The libarary is used to get the ASN numbers associated with IP addresses.This library is based on Maxminds GeoIP C API.
- urlparse :this is used to convert a relative url to an absolute url.
- Public suffix List :This is the collection of all registered host names given by all internet users.The Public Suffix List is an initiative of Mozilla, but is maintained as a community resource.It is available for use in any software, but was originally created to meet the needs of browser manufacturers.In our code we use it to get second level domain from a host name.The "effective_tld_namesdat" is the file which is free downloadable from their site.
- IPython :IPython notebook is used for pyspark code writting and execution.
- matplotlib :Python librarry used to for making different graphs used in this thesis.
- pygal :Pygal is also another python library which we used to make graphs.

4.3 Design of Crawler Engine

In this section,we will discuss the whole architecture behind the implementation.The design process involves two parts,first one is the architectural overview of crawler engine which takes 100,000 top ranked websites of alexa as input,crawl the websites , return result file as output from the engine and second part is processing of result file using data processing engine which internally use "pyspark".Along with this,we will also discuss little bit about "Scrapy framework" which is main backnone behind the crawler engine.The input to crawler and result output file matrics also will be discussed on the below section.

4.3.1 Crawler Engine

In this section we will discuss the internal architectutre behind the crawler engine.We will start with "Scrapy framework" which is the main crawler engine work behind the whole crawler engine.Scrapy framework is based on spiders which are self-contained crawlers worked by set of given instructions[1].As it is one of the top ranked open source project,most of the important documnets can be found online[5].But as we are going to use the terminologies to describe our crawler engine, very brief The main objective of crawler engine is to take input data in the format of text file which contain the top 100,000 websites of alexa.Then devide this master domain list into multiple sub domain lists with equal weighted websites which will be discussed in subsection 4.1.2 and then process each sub domain list using separate crawler instances which will be discussed briefly in below 4.1.2.

4.3.2 Scrapy Framework

Scrapy framework is one of top ranked open source project used for,a fast web crawling framework,used to crawl websites and extract structured data from their pages.It provide option for both focused and broad crawling.As we can see from figure-1,the main components of the framework contain Scrapy engine,scheduler,downloaders,spiders,item pipeline,downloader middlewares.

Scrapy Engine : The engine is responsible for controlling the data flow between all components of the system, and triggering events when certain actions occur. See the Data Flow section below for more details.

Scheduler : The Scheduler receives requests from the engine and enqueues them for feeding them later (also to the engine) when the engine requests them.

Downloader : The Downloader is responsible for fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders.

Spiders : Spiders are custom classes written by Scrapy users to parse responses and extract items (aka scraped items) from them or additional URLs (requests) to follow.

Item Pipeline : The Item Pipeline is responsible for processing the items once they have been extracted (or scraped) by the spiders. Typical tasks include cleansing, validation and persistence (like storing the item in a database). For more information see Item Pipeline.

Downloader middlewares : Downloader middlewares are specific hooks that sit between the Engine and the Downloader and process requests when they pass from the Engine to the Downloader, and responses that pass from Downloader to the Engine.

4.4 Scrapy data types

Scrapy uses some specific classes and strings which is used to crawl data.These are written in python and are open.Scrapy also use twisted framework for multithreading.

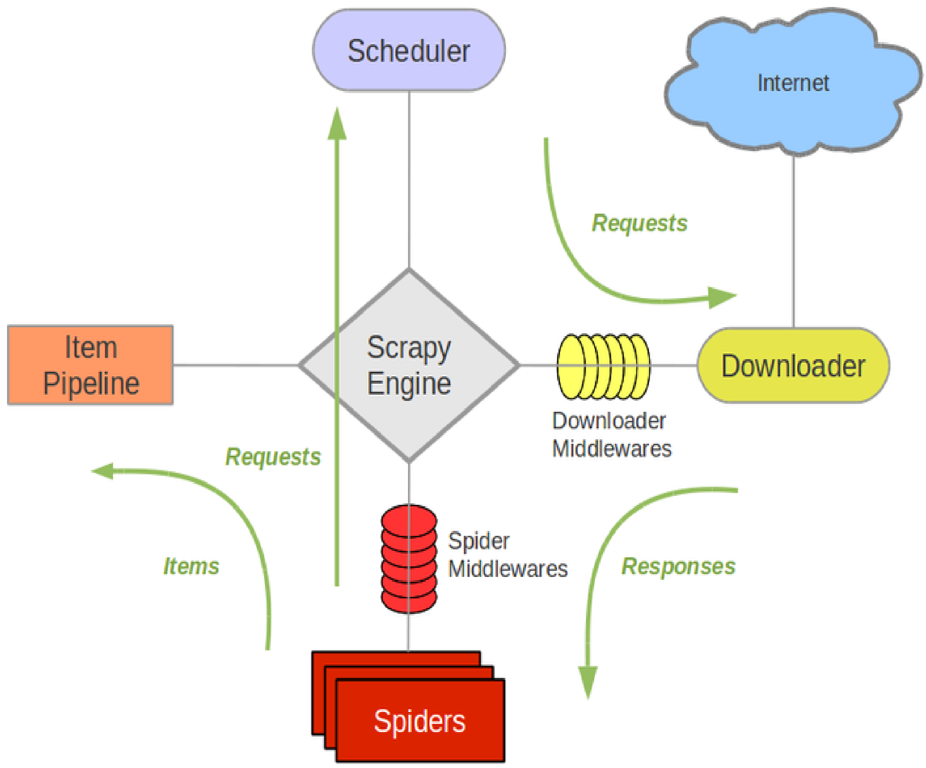

Figure 8: Scrapy Architecture

1. Spider :Spiders are the classes used to crawl single or multiple domains using different methods like start_requests(),parse().

   - start_requests() :This scrapy method is used to pass domains to parse method for further crawling process.
   - parse() :he parse method is in charge of processing the response and returning scraped data and/or more URLs to follow. Other Requests callbacks have the same requirements as the Spider class.

2. Twisted Framework:Twisted supports an abstraction over raw threads using a thread as a deferred source. Thus, a deferred is returned immediately, which will receive a value when the thread finishes. Callbacks can be attached which will run in the main thread, thus alleviating the need for complex locking solutions. A prime example of such usage, which comes from Twisted's support libraries, is using this model to call into databases. The database call itself happens on a foreign thread, but the analysis of the result happens in the main thread.

3. Requests objects:Typically, Request objects are generated in the spiders

and pass across the system until they reach the Downloader, which executes the request.A Request object represents an HTTP request, which is usually generated in the Spider and executed by the Downloader, and thus generating a Response.

4. Response objects :Rquest object returns a Response object which travels back to the spider that issued the request.A Response object represents an HTTP response, which is usually downloaded (by the Downloader) and fed to the Spiders for processing.

4.4.1 Equidivision of top websites

Crawler engine internally starts 20 crawler instances (check section4.1.2)and each instance take different site list as input.So we need to divide the master domain list into sublists but we also need to divide it such a way that all the crawlers will complete their crawling in almost same time.Hence we choose to create the sublists with equal weight.Weight points to rank of the websites here.

The master domain list contain the the website url and the rank of the websites.So after division the first sublist will contain the websites of 1st rank,21st rank,...99981st rank;second list will contain 2nd rank,22nd rank...99982nd rank websites and so forth for othre instances.Hence each sublist will contain 5000 websites with equal divison of website ranks.

4.4.2 Basic Architecture

The 100,000 top ranked websites of alexa is listed in master domain list.For better performance and to distribute load, we create multiple instances of crawler which run in separate threads. Each crawler create each own instance and run separate sublist of domains as input.The master domain lists to sublist mapping described in section 4.1.2.The figure 9 and 10 shows the basic architecture of crawler engine.Master domain list contain the top ranked websites of alexa. Each crawler instance work separately and save cralwed data into a output file.As all the crawler share the same output file we use python locking method so that at a time it can be used by a single instance.

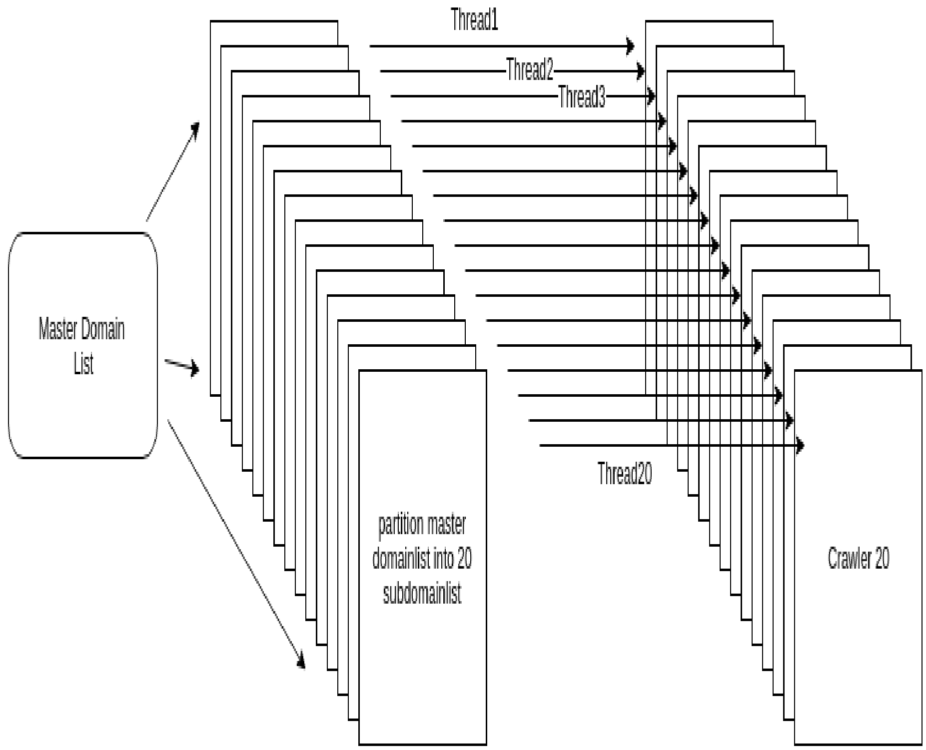

Figure 9: Basic Architecture

4.5 Conclusion

In this section we talked about internal architecture of crawler engine.Along with this we talked about all the tools,languages used to design the whole process.In th next section we will talk about the approach to collect traces, i. e., ac- tive DNS measurements, in order to evaluate our methodology.

5 Measurement

In this section we present our approach to collect various website information,in order to evalute our methodology.In order to acheive our goal ,we crawl a list of websites and analysis those data. This chapter explains the test experiment setup of the Scrapy framework and the data analysis framework . It will go into the hardware used, the setup of the software and the composition of the test series.

5.1 Hostname Selection

To obtain a good coverage of the largest hosting infrastructures, we crawl top 100,000 ranked websites from alexa.Alexa ranks websites based on internet traffic-users of its toolbar for various web browsers like Google Chrome,Internet explorer,Firefox. [6].

Moreover,websites contain a lot of embeded contents like images,videos, advertisements etc. that the browser of the user has to download from the various web servers.These web servers can be from differnt web content providers or from hypergiants. In our study, such embedded content has to be taken into account, as it might be served from servers other than those serving the front page of a popular hostname listed in top rank websites of Alexa.To give better understanding,while crawling facebook.com,the front page is serverd from facebook data centers while the logo and other meta data come from Akamai.For most of the websites,the important videos,images or other embeded contents present in the front page of the websites.So to make our study more precise,we crawl only the front pages of all the top ranked websites and the embeded links present in the front pages of those websites.

Overall we get total 219604 unique second level domains.But from them lot of SLDs which can be clubbed into other big SLD.the whole process of clubbing will be discussed in the following sessions.Once we get the total number of parent SLDs,we will take the top 2000 popular slds.

5.2 Measurement Approach

5.2.1 Data collected using Http Request

First we collect top 100,000 top ranked websites as of september 2016 from alexa website.The scrapy crawler queries the http get method to local dns resolver for all the websites and store the results in a trace file.In this way we are able to crawl total 85,000 top ranked websites of the alexa which resulted total 13919464 traces.Each trace contain total 10 different data which are described below.

1. index :index shows the rank of the website
2. depth_level : 0 or 1. 0 shows that the website is the main page url and 1 shows the embeded links inside main page
3. httpResponseStatus :the http return status code.

4. MIMEcontentType :this is included to know the type of element inside the web page.

5. content_length :content length gives the idea about the size of the element.

6. url :this is the url to be crawled by the scrapy engine.This url can be main page url or embeded links.Duplicate links are omitted for the same main page by using RFPDupeFilter.RFPDupeFilter is a scrapy class which is used to detect and filter duplicate requests [3].

7. cookies :cookies involved in a website

8. tagType :this shows the html element type. for example if an element is embeded in a wensite like ”¡img class=”desktop” title=”” alt=”” src=”img/bg-cropped.jpg”¿”,then we collect the tagtype which is img here. ARecord=This contain all the aRecord names involved in a website while resolving to the ip address.

9. destIP :this column stores the corresponding resolved ip addresses of the website.For example,for the website bmw.com (figure-6),the destIP will store 72.247.184.130 and 72.247.184.137

10. ASN_Number=Field():This column stores the ASN number of a IP address.For this we use maxmind IP to ASN mapping file.

11. distinctASNs=This will keep the total number of ASNs involved for all the embeded website links for a perticular mainpage url which is one of the alexa’s top ranked website.

12. ObjectCount=Field():This column stores the total number of external links embeded in a website.

13. NumberOfuniqueExternalSecondlevelSites :This columns gives the toatl number of unique second level domains involved in all the embeded links of a website.

14. start_time :This contains the staring time when the http starts requesting for the website.

15. end_time=This contains the staring time when the http ends requesting for the website.

5.2.2 Data collected from RIPE RIS

After whole crawling process over for 100,000 top ranked websites for alexa,we collected total 150,000 unique IP addresses.To get the

5.3 Test Enviornment

For setting up crawling server enviornment ,we used one server machine of 1 Gb link .The hardware configuration of the server can be seen below.

	Server
Processor	Common KVM processor
CPUs	6
CPU Mhz	2400
Cache size KB	4096
RAM KB	8194900
Operating Sytem	Ubuntu 14.04.2 LTS
Kernel	3.13.0-46-generic

Our crawling infrastructure is based on Scrapy.For the scrapy setup we are using scrapy 0.24.6 (Released date 2015-04-20)[5] and python 2.7.6 .

analysis standpoint (.csv, .txt, etc.). - In addition to the HTML-encoded tabular data, for each output entry we would like to add the information encoded in the URL, in the form of a standardized date stamp as well site and region ID numbers. - While a detailed folder structure is not necessary, file nomenclature is critical; con- sidering the volume of data points, for each site and each timescale, one output file should be generated for each years worth of data, and the file naming should reflect region, site, timescale, and year. - Again due to the volume of data, as a bare minimum, given a certain timescale the program should generate all the required output for a single site over the entire span of interest. - Ideally, since we may not need data from every site, given a timescale we would like to give our program a list of sites to visit, and have it au

5.4 Number of threads Selection

5.5 Data Cleanup

We perform a thorough cleanup process on the raw traces.In each crawling we gather 16 different metrics listed in section 4 (Design and specification).Hence we pass each crawled link into regular expression to check the validity of the links based on number of metrics,type of matrics.

Through scrapy we crawled total 91,398 sites out of top 100,000 websites of alexa website which give us total 13,306,791 website links.After removing all traces with the above artefacts, we have 13,306,790 clean traces that form the basis of this study Note, the cleanup process has limited impact on our hosting infrastructure coverage and sampling of the network footprint.

5.5.1 Data Analysis Setup

For data cleaning up we are using pyspark 1.5.2 alsong with python 2.7.

	Server
Processor	Common KVM processor
CPUs	6
CPU Mhz	2400
Cache size KB	4096
RAM KB	8194900
Operating Sytem	Ubuntu 14.04.2 LTS
Kernel	3.13.0-46-generic

5.6 Conclusion

6 Results

In this section,we will identify the prominent hosting infrastructures,categorization of hypergiants and lastly study their role on other web sites.Most content delievery infrastructures can be found in the home page of any website.Hence we keep our crawling process to only two depths which mean we crawl top 100,000 websites from alexa website and the embeded links assosciated in their home pages.

6.1 Presence of prominent infrastructures

In this section,the presence of of prominent infrastures will be determined first and then will catagories the top 20 CDN hosts according to their presence in all other web links.Overall, the output of our algorithm leads to the identification of more than 219,604 potentially distinct SLDs.Figure-8 shows, for each SLD infrastructure, the number of urls serverd by the SLD infrastructure, on a log-log scale. The SLD infrastructures are ranked in decreasing order based on number of urls hosted on those infrastructure.There are two measure points can be taken out of this graph.

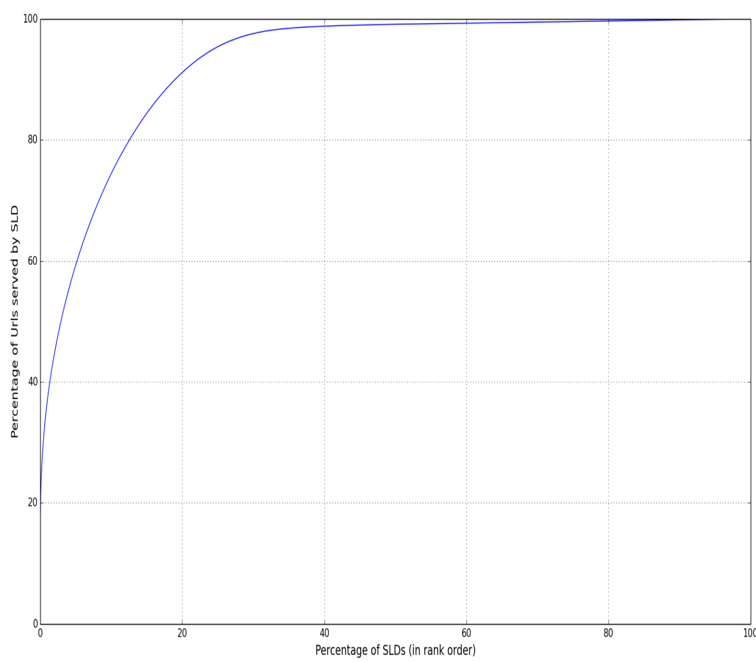

Figure 10: Percentage of SLDs vs Number of urls served

Firstly,most of the hosting infrastructure serve a single hostname. From figure-8,this is observed that nearly 53,44% of hosting insfrastructures serve only single url.While this may lead one to conclude that there is a one-to-one correspondence between the SLD and site which could be because of size of website or content of the website distributed between different SLDs.In most of cases these hosts likely to be located in a single facility and serve non-replicated content.

Second,a few SLD infrastructures are serving a large number of urls.In fact, from our data,the first 20% SLDs hosting almost 91.13% of all urls crawled and top 10% of SLDs hosting almost 74.39% of all urls.Even first 1% of SLDs serving 36.82% of urls.This higher percentage of urls implying presence of content delievery networks as well as content producers.The resulting data allows us to make qualitative observations, namely that well-known hosting infrastructures are represented.

6.1.1 Conclusion

In this section, we check our data according to Zipf's law and we found that top 20% of SLDs serve almost 92% of urls which evident of Zipf's law in today's interenet traffic.Again we found that first 1% of SLDs serve alomost 37% of urls which implying the presence of prominent CDNs and hypergiants in top 1% or 2196 SLDs which is will be used in next section to find the hypergiants.

6.2 Prominent infrastructures and why we need clustering step ?

In this section we will identify the prominent infrastructures present in internet based on number of urls served by them.The brief description will be provided.In the subsequent session we will discuss the necessity of clustering method.

6.2.1 Identification prominent infrastructures

The top 20 SLDs serve almost 13.13 % of all urls crawled.These 20 SLDs contain not only CDNs like akamai,cloudflare etc,it also contain other slds which are responsible for the sites which are predominantly contain hugh amount of content like facebook,twitter etc.It also contain slds like amazonaws which are provide cloud hosting services and some serving hosting web hosting companies like ccgslb.net.Hence these top 20 SLDs are giants in their fields.Hence we need to discuss little bit on who are these SLDs and how they are operating in Internet based on their role.

1. Content Delievery networks :

   Akamai tops the list with two of its slds take the first two position in the list.The slds are akamaiedge.net and akamai.net.Akamaiedge.net carries 2,56% of all the website links from the total crawled links where as akamai.net hosts 2,27% of urls.Hence in total akamai carries almost 5% of all the urls which is hugh when we talk about 219,604 potentially distinct

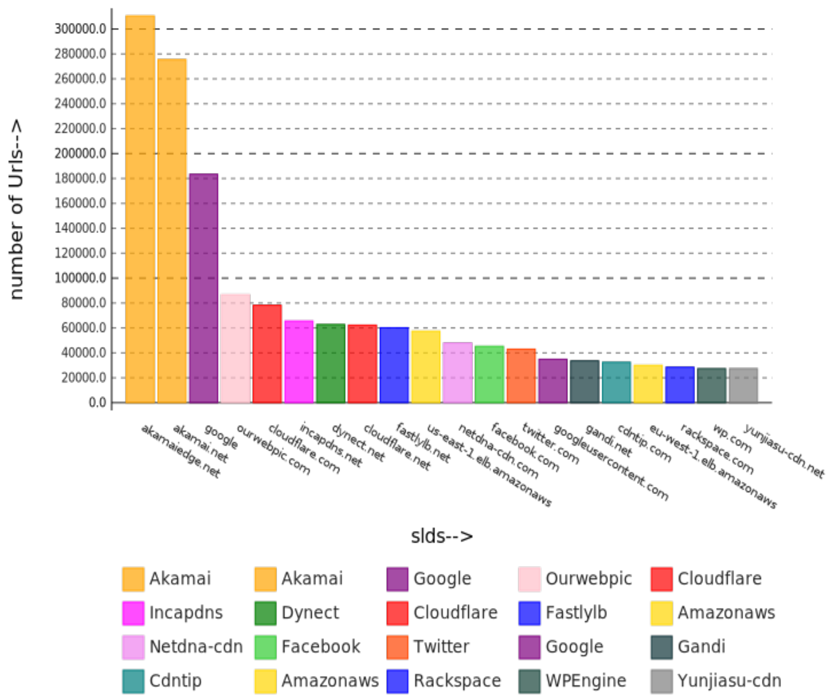

Figure 11: Top 20 SLDs

SLD infrastructures.This can be because of its broad ranges of commercial services beyond content delivery,delivery of live and on-demand high-definition (HD) media, high-availability storage, analytics etc.Comprising more than 61,000 servers located across nearly 1,000 networks in 70 countries worldwide, the Akamai platform delivers hundreds of billions of Internet interactions daily, helping thousands of enterprises boost the performance and reliability of their Internet applications [2].

Cloudflare is one of the top CDN companies in the world.In top 20 slds we have three cloudflare slds cloudflare.com,cloudflare.net and Yunjiasu-cdn.Cloudflare.com,cloudflare.net,and Yunjiasu-cdn serves 0,64 %, 0,51 % and 0,22 % of all website urls repectively.It provides a various services staring from content delivery network,Internet security services and distributed domain name services.Yunjiasu is cloudflare cdn use in China.Baidu Yunjiasu is the first-ever platform available in Mandarin to make Internet applications fast, secure, and more intelligent in China and around the world. Yunjiasu offers the same features and functionality as CloudFlare (CDN, DNS, DDoS protection, WAF, etc), optimized for performance within China, and tailored for the China market[https://www.cloudflare.com/china/].

ourwebpic is used for HICHINA ZHICHENG TECHNOLOGY LTD which is one of the biggest Wholesale supplier in International Online Shopping offers Electronics, Apparels, Home and Garden, Gift etc to customers from China.

2. Content Providers : Similar observations are made for google.According to Figure-9,google.com carries 1,51 % of all website links.Similarly googleusercontent.com serves 0,28 % of urls. This is due to google's various services like google web search ,youtube pages which are hosted on "youtube-ui.l.google.com". Except the countrywise google host domains,google also use other infrastructure for different purposes.For example 'earnmoneywithgoogleadsense.com', 'googleads.com' are used to provide the advertisement service to customers. Along with that google also provide google global cache servers which are placed in different ISP's infrasturucture. Google Global Cache (GGC) allows a large portion of these requests to be served from a small node located inside the service providers network serving regional users [3].

3. Consumer Content : Facebook.com and twitter.com are two major player in top 20 giant SLDs list.They server almost 0,37 % and 0,35 % of urls respectively.It is because of vast use of social media sites in daily life.Again lot of other sites embeded the content of facebook and twitter in their site.

4. Hosting Companies : We can find major SLDs from figure-9,one is CCGSLB.NET nad incapsdns.net.Bothe are registered under Godaddy.com which mainly provide web hosting service.Dyn DNS Company (pronounced "dine") is an Internet performance management company, offering products to monitor, control, and optimize online infrastructure, and also domain registration services and email products.

5. Cloud Computing CDNs : Rackspace Inc. is a managed cloud computing company based in Windcrest, Texas, USA, a suburb of San Antonio, Texas. And Amazonas

From figure-9 we can also deduct that these companies can be treated as hypergiants in their own field.But only serving hugh amount of links might not be only sufficient criteria instead we need to check how they are distributed all over world by counting number of bgp prefixes they routed to.How many autonomous system they have.This will provide a better picture on deducting hypergiants.

6.2.2 why we need clustering step ?

From figure-9 ,it is found that there are some slds which are served by same company.Like akamaiedge.net and akamai.net both are CNAMEs used by Akamai company.Similarly google.com,googleusercontent.com,googlehosted.com and googledomains.com all are used by google.Normally companies used different names when they serve different services to the customer or different name for certain located customers.This can be seen from the CNAMEs used by amazonaws.us-east-1.elb.amazonaws.com,eu-west-1.elb.amazonaws.com are two examples of naming pattern used by the amazon to provide services to dictinct located customers.But here the question arises these names are pointing to same infrastructure or different.If they are pointing to same infrastructure they should be considered as one and else separately.To measure this we are going to take RIPE bgp prefixes each SLD routed to and cluster all the SLDs which use the same bgp prefixes.

6.2.3 Conclusion

This is clear from the section that there are large SLDs which are contribute to maximum traffic in the internet but can we take all the prominent CDNs as hypergiants.From figure-9 we can also deduct that these companies can be treated as hypergiants in their own field.But along with this we can infer below observations.

1. First one is there are some SLDs which are served by same parent company like akamaiedge.net and akamai.net.Hence are they same or different SLDs ?

2. To get the hypergiants only classfying SLDs based on number of urls are not sufficient crieteria instead we need to check the footproint covered by these companies in the world by checking their bgp prefix routes.

In the next section,we will discuss about who are these hypergiants and our method to classfy those.

6.3 Classification of hypergiants

In 2010, Craig Labovitz, then of Arbor Networks,characterized the hypergiant as a content provider that makes massive investments in bandwidth, storage, and computing capacity to maximize efficiencies and performance.But as the architecture of internet evolves,researchers found that the internet has now become a flatter infrastructure where there are fewer autonomous systems connected to each other and they try to have a bigger footprint in terms of number of bgp prefixes than before.In this way they are able to diversify their architecture as well as able to move content to even closer to their customers.They termed this intrastructure providers as hypergiants.[Atos White Paper].

In last section from figure-9,we find some prominent infrastructures in internet but we also observed that there are some slds which are from same company like akamaiedge.net and akamai.net both are from same company Akamai.Similarly google.com,googleusercontent.com are from the company Google.Hence there is necessity of clustering the same slds together.But this clustering,we follow our clustering algorithm which described in chapter-2 methodology section.In this clustering algorithm we will add the RIPE bgp prefixe routes resolved from IP addresses.

To classify these prominent SLD infrastructures into hypergiant group we will take top 2175 large slds which represent top 1% of sld and serve also 36 % of urls.According to our observation,we should able to get hypergiants from this number and classify their network coverage all over the world.

In this section first we will discuss about the cluster algorithm validation and then we will see who are these top hypergiants.

6.3.1 Cluster Validation

As we can see from

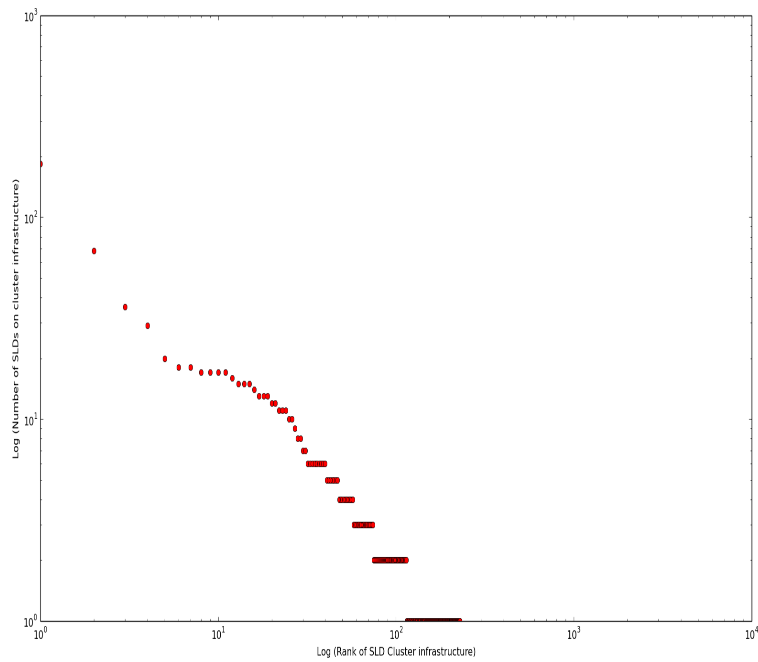

Figure 12: Number of SLDs served by different SLD infrastructure clusters.

6.3.2 case 1 :number of hosts Vs number of IP addresses

we extract the IP addresses obtained within the DNS replies. The set of IP addresses returned for a particular hostname reveals the degree to which the corresponding hosting infrastructure is network-wise distributed.The host name having a lot of IP addresses mean having served by a lot of host servers which indicate the network-wise distribution of the hostname.

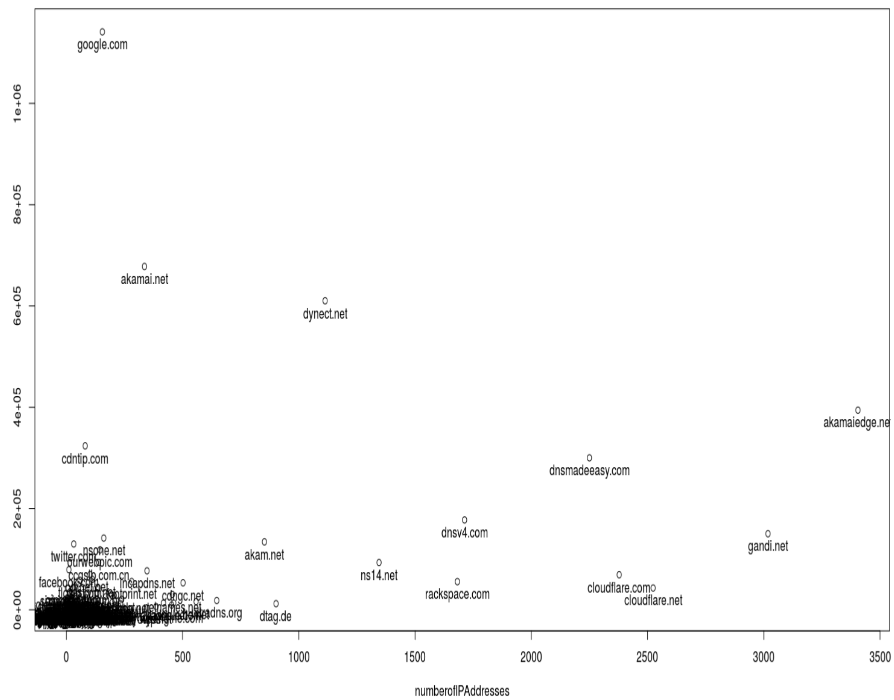

Figure 13: Scatter plot -number of hosts vs number of unique IP addresses

We can cluster the figure-4 into 3 parts.First one is google.com which is showing maximum number of links but very less number of IP addresses ,which might be because of globally distributed edge caches to cache HTTP(S) load balanced content close to your users like Google cache.more cache servers near to the crawling server.

Second cluster can be of less number of hosts but having a lot of IP addresses like cloudflare.net,cloudflare.com.By default, CloudFlare's CDN does not cache HTML content. HTML updates will show immediately. Only static content like Javascript, images and CSS would be cached by default.

Third cluster can be of more links and more ip address.Here akamaiedge,net is hsoing that.This shows that the content are distributed different parts of world.

6.3.3 case 2 :number of hosts Vs number of IP subnets

/24 subnetworks have the advantage of better representing the actual usage of the address space by highly distributed hosting infrastructures such as Akamai.

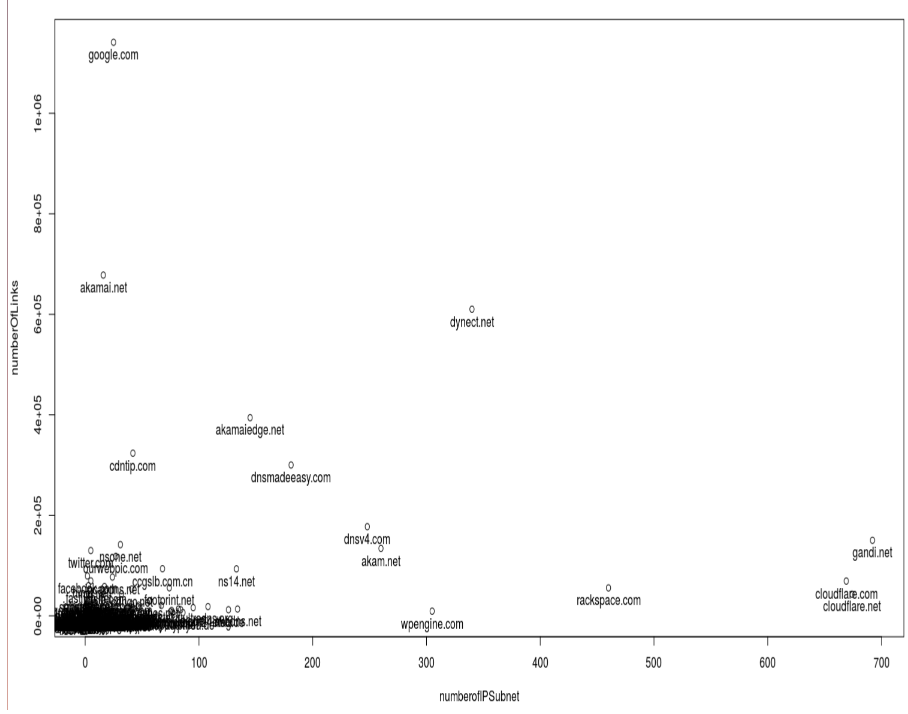

Figure 14: Scatter plot -number of hosts vs number of unique 24 subnet

From figure-5 ,we can cluster whole hosts into 3 parts: high links but less IP subnets:like google.com.Its because of google cache servers.

Second cluster can be of less number of hosts but having a lot of IP addresses subnet like cloudflare.net,cloudflare.com.By default, CloudFlare's CDN does not cache HTML content. HTML updates will show immediately. Only static content like Javascript, images and CSS would be cached by default.

Third cluster can be of more links and more ip address.Here akamaiedge,net is hsoing that.This shows that the content are distributed different parts of world.

6.3.4 case 3 :number of hosts Vs number of IP subnets Vs number of IP addresses

The case 1 and case 2 ,both are showing as simliar kind of results.Hence to get better idea we will take all features into account and try to cluster hosts which can be taken as hypergiants.

The figure 6 can be clustered into several culsters. (high links,high ipsubnet,high ipadd) If we take all the second level domains (akamai.net,akamaiedge.net,akam.net,akadns.net,akamai)

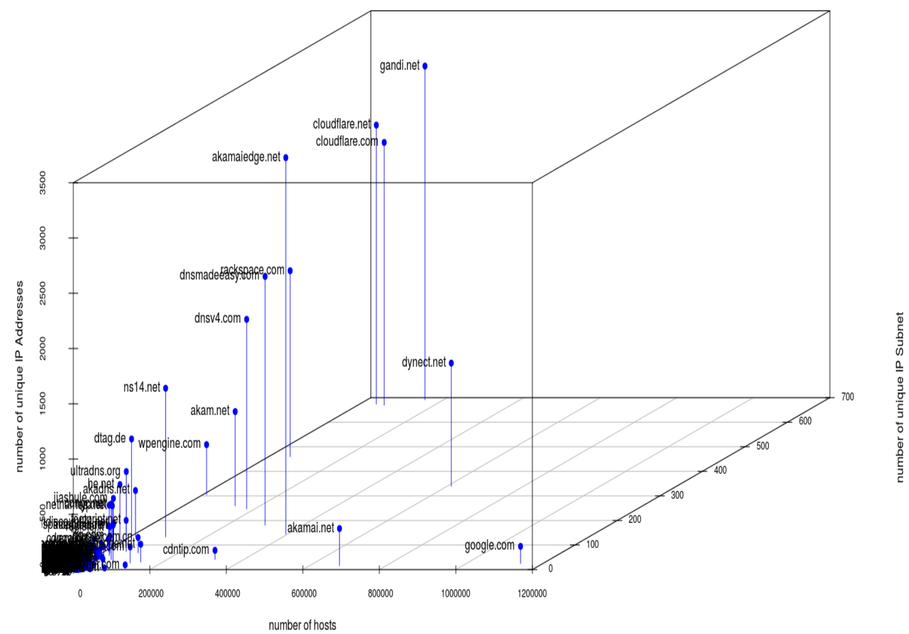

Figure 15: 3D scatter plot taking all features

(less links,high subnet,high ip) gandi.net ,cloudflare.net ,cloudflare.com :This shows

(high links,less ip,less ipsubnet) google

6.3.5 Conclusion

From the ablove section,we can conclude that akamai,google,cloudflare for its wide range of presence,use of vast infrastructures can be considered as hypergiants. Next section we will see the effect of these hypergiants on other websites.

7 Future Work

8 Conclusion

9 Appendix

References

[1] T. Leighton. *Improving Performance on the Internet.* Commun. ACM, 52(2):4451, 2009.

[2] C. Labovitz, S. Lekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. *Internet Inter-Domain Traffic.* In Proc. ACM SIGCOMM, 2010.

[3] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann. *Improving Content Delivery using Provider-Aided Distance.* Information. In ACM IMC, 2010.

[4] G. Maier, A. Feldmann, V. Paxson, and M. Allman. *On Dominant Characteristics of Residential Broadband Internet Traffic.* In Proc. ACM IMC, 2009.

[5] Nygren, R. K. Sitaraman, and J. Sun. *The Akamai Network:A Platform for High-performance Internet Applications.* Syst. Rev., 44:219, August 2010.

[6] R. Krishnan, H. Madhyastha, S. Srinivasan, S. Jain,A. Krishnamurthy, T. Anderson, and J. Gao. *Moving Beyond End-to-end Path Information to Optimize CDN Performance.*

[7] *https://atos.net/content/dam/global/ascent-whitepapers/ascent-whitepaper-internet-evolution.pdf.*

[8] Schonfeld, E. *Eric Schmidts Gang Of Four: Google, Apple, Amazon, And Facebook.TechCrunch.* Retrieved from http://techcrunch.com/2011/05/31/schmidt-gang-four-google-appleamazon-facebook/

[9] Bernhard Ager,Wolfgang Mhlbauer,Georgios Smaragdakis,Steve Uhlig *Web Content Cartography.*

[10] Bernhard Ager *Impact of Location on Content Delivery.*

[11] Yuval Shavitt, Udi Weinsberg. *Topological Trends of Internet Content Providers.* SIMPLEX 12: Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners (2012): 13-18.

[12] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, kc claffy, George Riley *AS Relationships: Inference and Validation.* SIGCOMM Computer Communications Review 37, no. 1 (2007): 31-40.

[13] Manuel Palacin, Miquel Oliver, Jorge Infante, Simon Oechsner and Alex Bikfalvi *The Impact of Content Delivery Networks on the Internet Ecosystem.* Journal of Information Policy, Vol. 3 (2013), pp. 304-330

[14] Mike Axelrod *The Value of Content Distribution Networks and Google Global Cache.*

[15] http://www.hpl.hp.com/research/idl/papers/ranking/adamicglottometrics.pdf.
.

[16] *http://doc.scrapy.org.*

[17] *"Technology: How and Why We Crawl the Web". Alexa. Archived from the original on April 2, 2014. Retrieved November 6, 2011..*

[18] *Domain Name System* . https://www.ietf.org/rfc/rfc1034.txt

[19] K RISTOL , D., AND M ONTULLI , L. *HTTP State Management Mechanism.* RFC 2109.

[20] K RISTOL , D., AND M ONTULLI , L. *HTTP State Management Mechanism.* RFC 2965.

[21] Lada A. Adamic ,Bernardo A. Huberman *Zipfs law and the Internet* . Glottometrics 3, 2002,p 143-150

[22] *RIPE Routing Information Service.* http://www.ripe.net/ris/.