**Network Architecture - 1**

**Project-2**

**Analysis of Inter-Domain Routing Data**

**By,**

**Name : Swati Soni**

**Name : Vineela Atluri**

**Name : Krishna Kumari D.**

1. **Project Team**

|  |  |  |
| --- | --- | --- |
| **Name** | **Id** | **Role** |
| **Swati Soni** | **16107465** | **Project Leader** |
| **Vineela Atluri** | **16103576** |  |
| **Krishna Kumari D.** | **16106929** | **Paper Reader** |

1. **Objective**

To analyze the inter domain routing data of BGP by using a scripting language like Perl for data analysis for better understanding of the BGP routing data and understanding the various trends in the data over a period of time.

1. **Answers with procedure, Code and comments:**

**Answers to the questions by using 6 rib files.**

**I followed a sequence of steps for obtaining acsii files:**

**The First Step is to Download Data from routeview.org and convert it into ascii format. This is shown in the following steps and screenshots.**

**BGP Data files used are :**

1. **rib.20111001.0000.bz2 (Default) for primary purpose.**
2. **rib.20110701.0000.bz2**
3. **rib.20110801.0000.bz2**
4. **rib.20110901.0000.bz2**
5. **rib.20111101.0000.bz2**
6. **rib.20111201.0000.bz2**

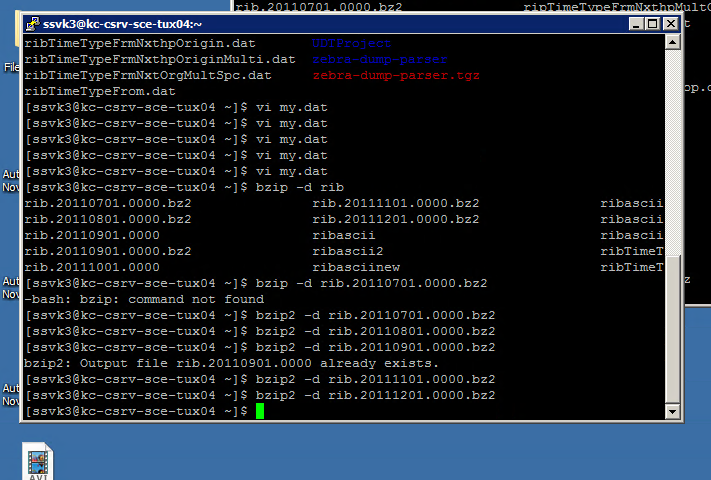


Figure 1. Unzipping the data

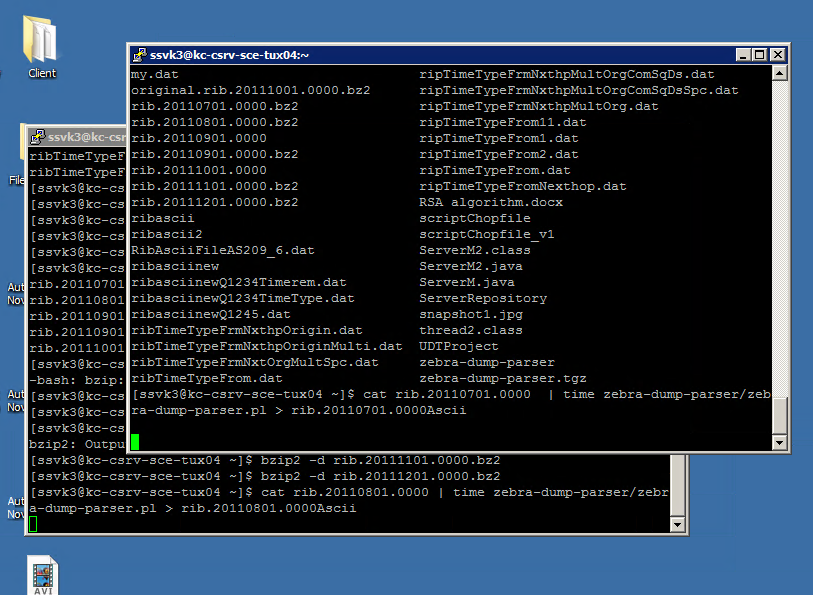


Figure 2 : Using Zebra parser for parsing files into ascii format.

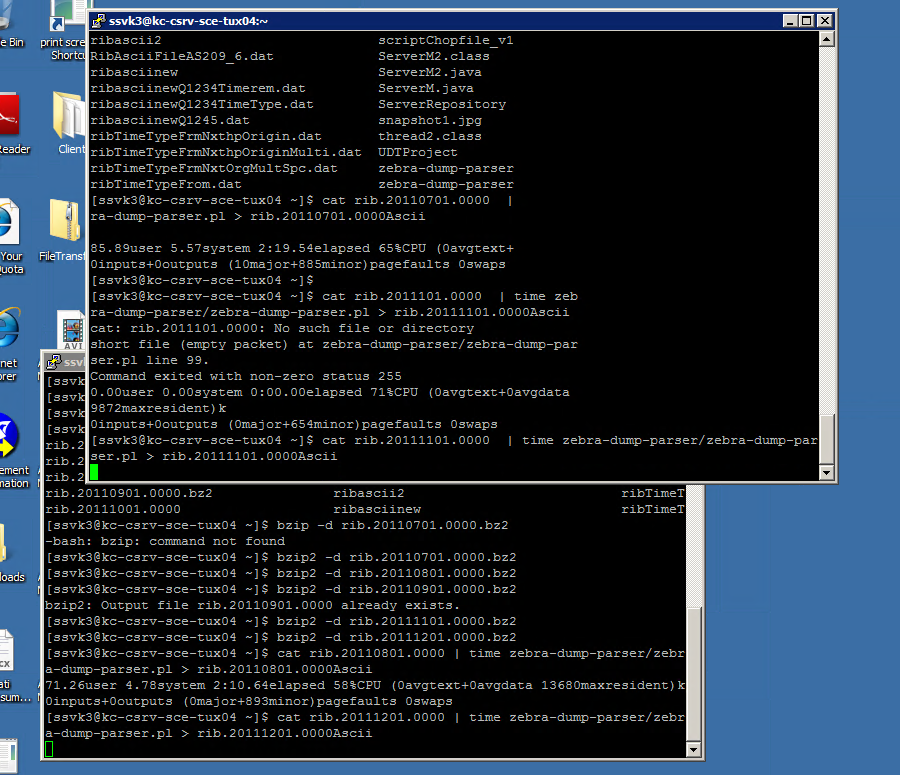


Fig 3. Further Conversion of remaining files into proper formats.

**Now consider the following Questions:**

**Answers to the 2,4 and 5 questions are provided in the section Right after Question 1. Then I have shown Q3 ,Q6 and Q7 answers:**

1. **How many different prefixes appeared in BGP data?**

Ans : 389796

**Procedure**

**Step 1 :** Let me first show the Number of Prefixes in 4920000 lines of of ascii BGP data:



This value is 71877 and time taken to compute the prefixes is 988 seconds.

This analysis give me a some picture of how the analysis of the rest of data going to be.

Now consider the entire BGP file and the following screen shot:



**Now the value of Total Distinct prefixes is : 389796 .**

The computation is performed in time : 2256 seconds. == 37.6 minutes.

**Code : Following is the perl script for analysis of the BGP data.**

#! /usr/bin/perl -w

while(<>){

#while(/(string(\d{1,3}))/g){

# print "$1\n" if $2 <= 100;

# }

}

#FH1 = open "abc.txt";

#open(FH1,"outputRib.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFileAS209\_6.dat")|| die "can not open the file \n";

#@abcfile = <FH1>;

$wordPrefix ="PREFIX";

$wordAS\_Path ="AS\_PATH";

$count=0;

$subnetStr="";

$preSubnet="";

$nextSubnet="";

$distinctPrefixes=0;

%hashASTable ;

#start Time

$start = time();

#@FilesToRead = ("outputRib.dat");

@FilesToRead = ("ribfinal4920000I.dat","ribfinal4920000II.dat","ribfinal4920000III.dat","ribfinal4920000IV.dat","ribfinal4920000V.dat","ribfinal4920000VI.dat");

foreach my $p (@FilesToRead)

{

open(FH1,$p)|| die "can not open the file \n";

@abcfile = <FH1>;

foreach $line(@abcfile){

#print "Line : $line";

$prefixIndex = index($line,$wordPrefix);

if($prefixIndex>=0){

print $line;

#($name,$subnet) = split("\s",$line);

# print "Name : $name\t"."Subnet : $subnet";

my @values = split('\s',$line);

# foreach my $val (@values){

# print "$val\n";

# }

($first,$value) = split('\s',$line);

if($count==0){

$preSubnet=$value;

}else{

$nextSubnet=$value;

}

if($preSubnet ne $nextSubnet){

#print "Pre subnet :$preSubnet \t"."next Subnet: $nextSubnet";

$distinctPrefixes++;

$subnetStr=$subnetStr."\n".$value;

if($count>0){

$preSubnet=$nextSubnet;

}

}

$count++;

}

}

print "Closing file FH1";

close(FH1);

}

# end timer

$end = time();

print "\nDistinct Subnets: $subnetStr\n";

# report

print "\n\nTime taken was ", ($end - $start), " seconds";

print "\nNumber of Distinct prefixes :$distinctPrefixes";

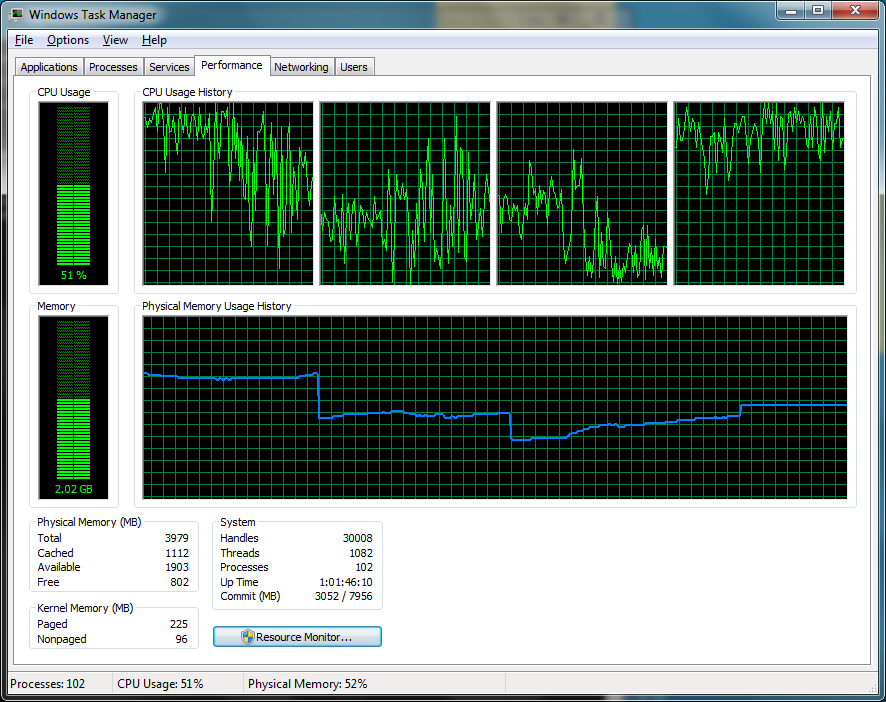
1. **How many different Autonomous Systems appeared in BGP data? Shown in the screenshots, explaination , procedure and code below**
2. **What is the average/maximum degree of AS connectivity? Shown in the screenshots, explaination , procedure and code below**
3. **Plot the distribution of degree of AS connectivity. Shown in the screenshots, explaination , procedure and code below:**

I decided to work over chopped data for the above 3 questions . The reason is performance and system memory utilization concerns for analysis.

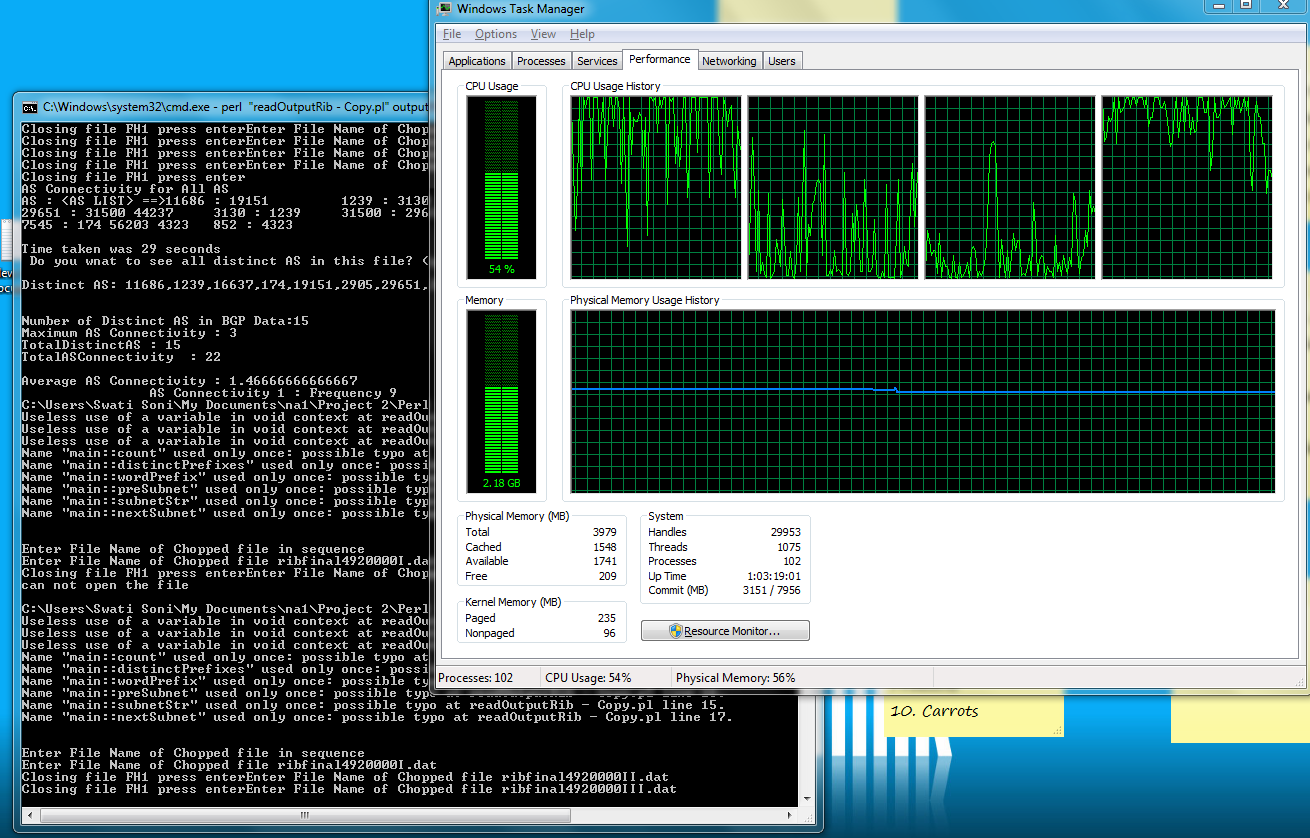
Everybody needs a solution but not at the cost of one’s own system (especially the one in which is your personal system containing all files and folders.)

So I decided to work on files having 4920000 lines of code. Why I chose this particular size is explained in this section “Trick for handling large volumes of data” , later in the report.

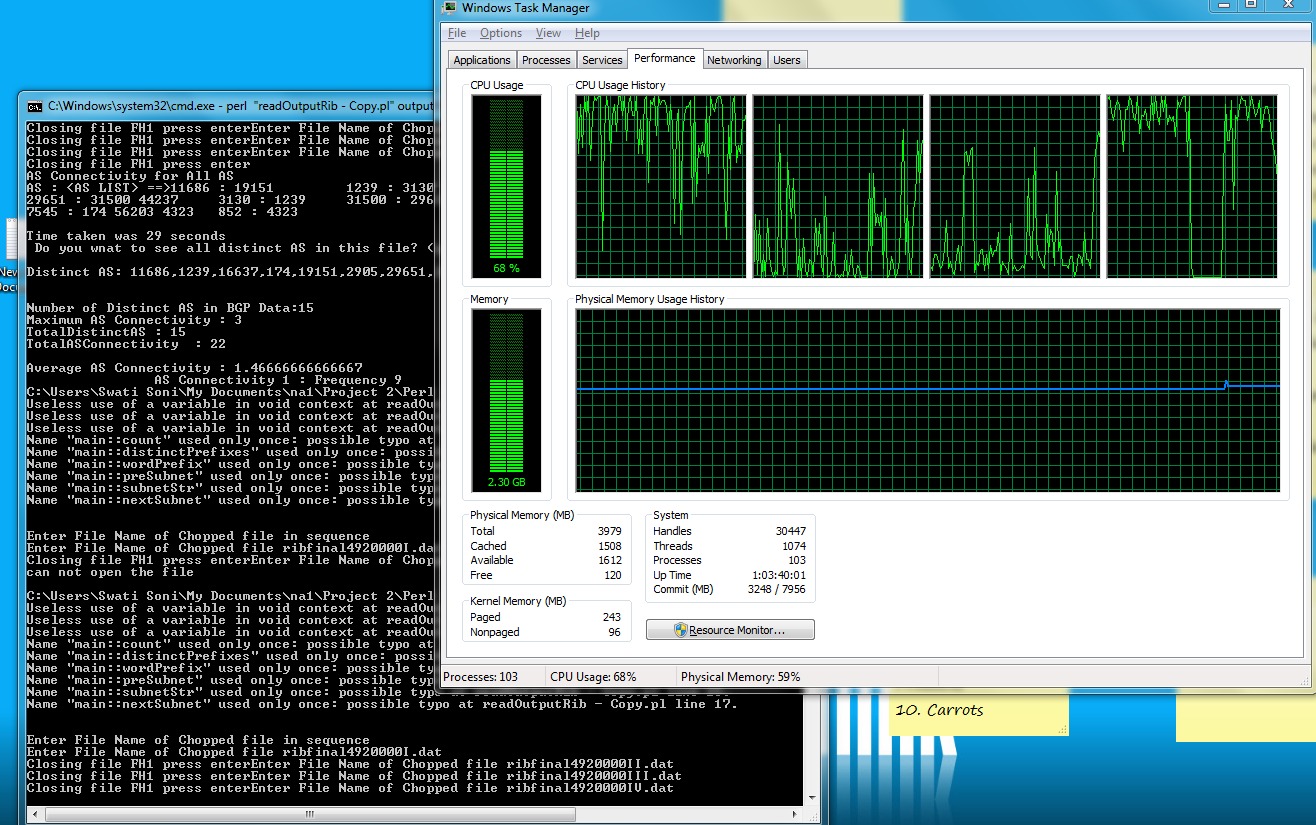
Also while I input the chopped files one by one into the system , I observed the trends on my system’s memory utilization and performance. These datas can be used for any future analysis on my codes or further additional work.



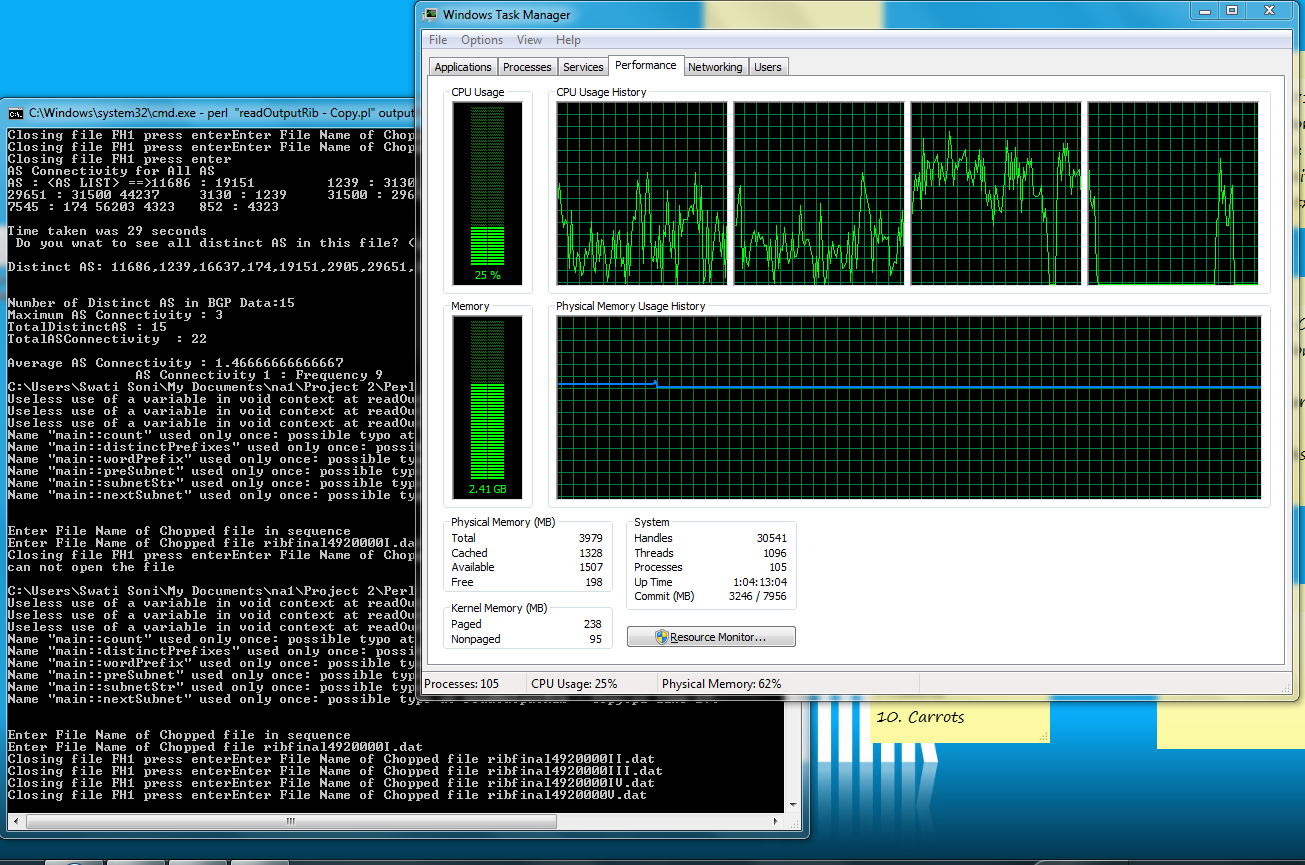
**Fig : System Performance measurement just before reading of First chopped file : indicated by the last step up in physical memory utilization. : 2.02 GB.**



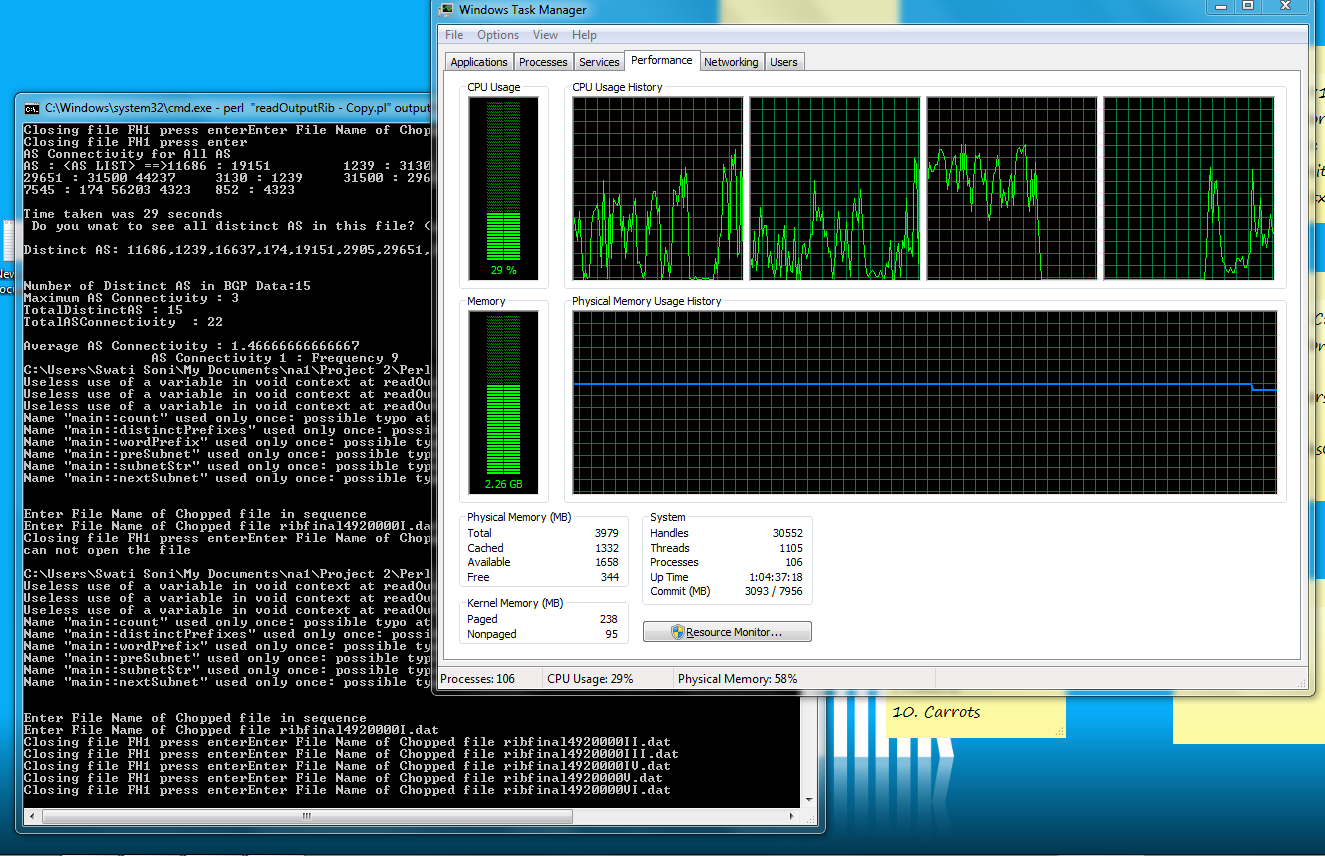
**Fig : system performance and memory utilization while analysis of 3rd chopped part of the BGP Ascii file.**



**Fig : system performance and memory utilization while analysis of 4th chopped part of the BGP Ascii file.**



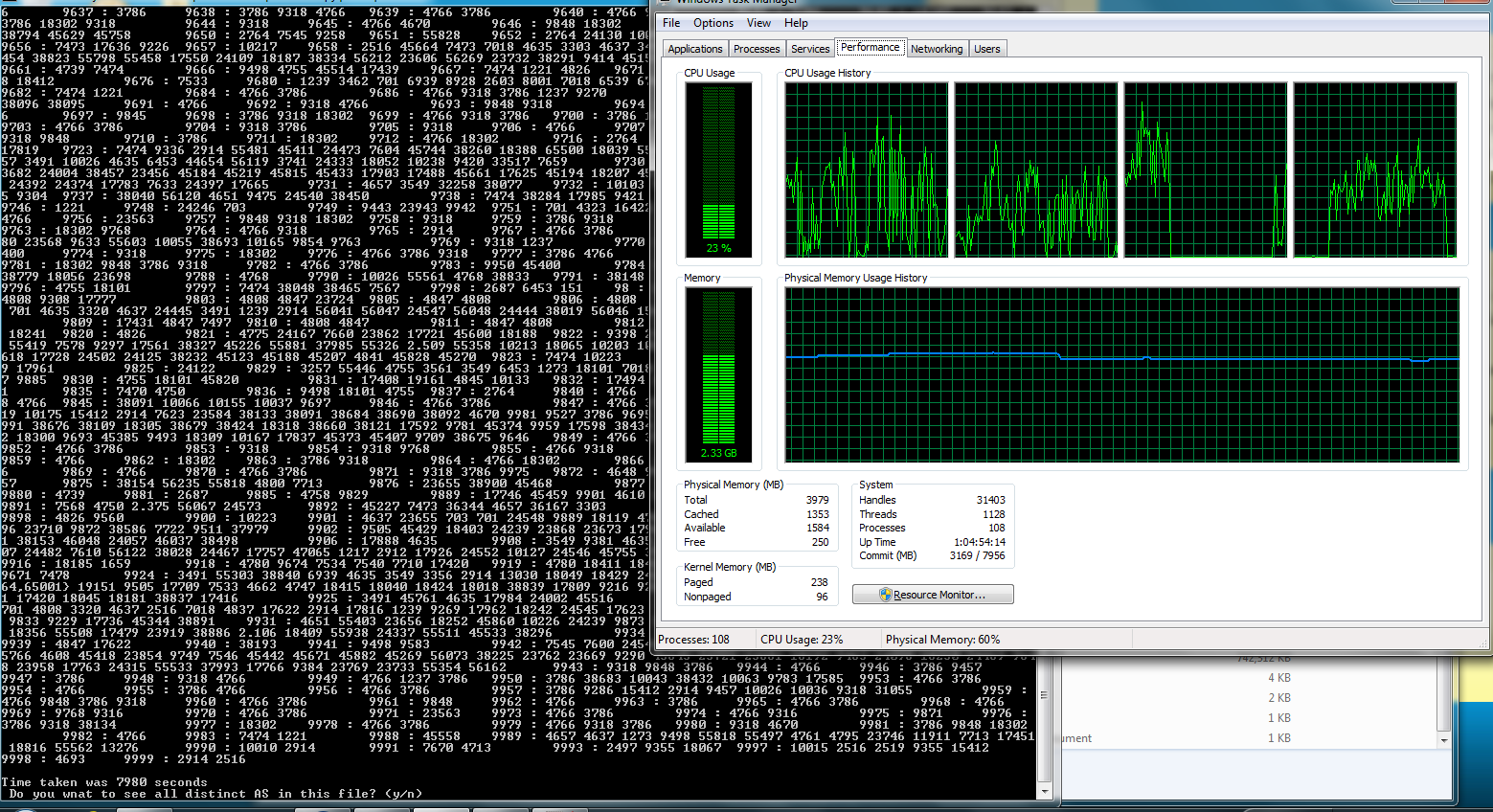
**Fig : system performance and memory utilization while analysis of 5th chopped part of the BGP Ascii file.**



**Fig : System performance and memory utilization while analysis of 6th chopped part of the BGP Ascii file.**

**Notice the ditch in memory utilization : last chopped files is smaller then other chopped files : GOOD NEWS >> Result is due soon.**

**Finally the result Screen Shot :**



**Fig : System performance and memory utilization Right after the analysis**

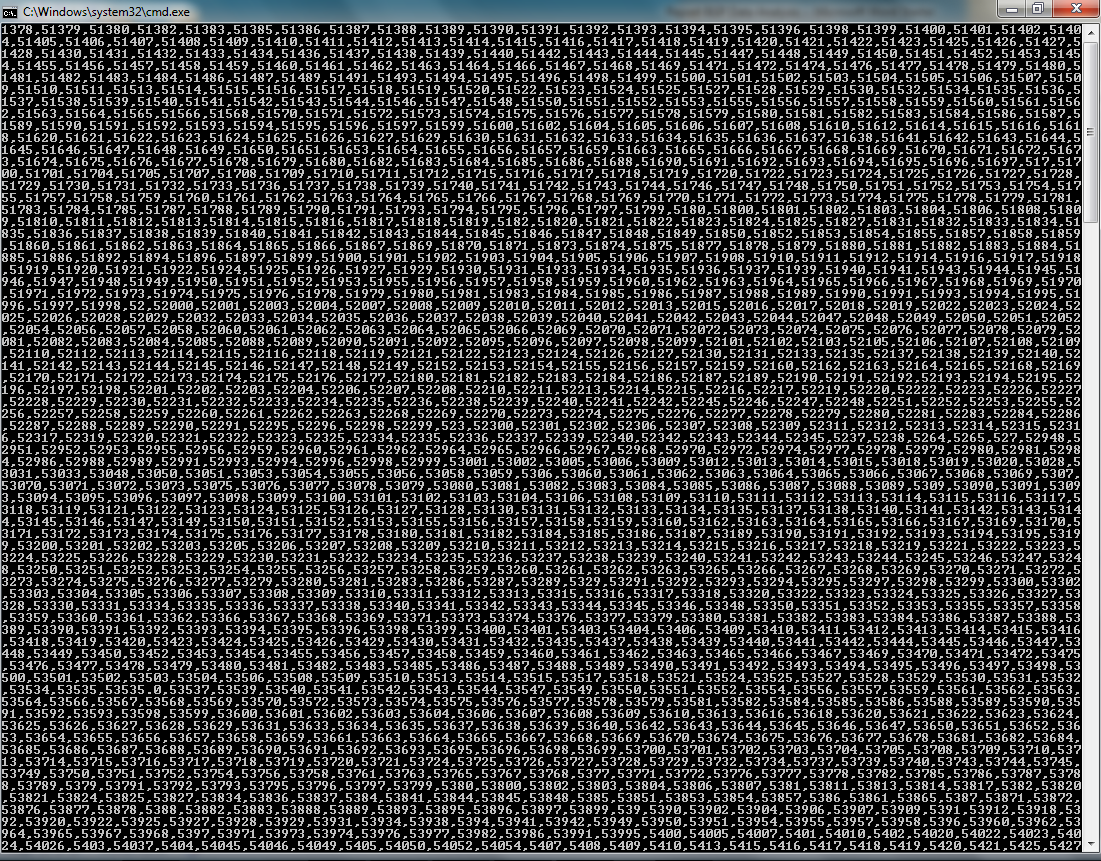
**Before the “Time Taken was 7980 seconds” line is mentioned maps of AS -> <connected AS> .This data is used for analysis of 2,4 and 5th Questions.**

**Overall time taken : 7980 seconds = 133 minutes = 2.21 hrs. (Very good performance over reading datas).**

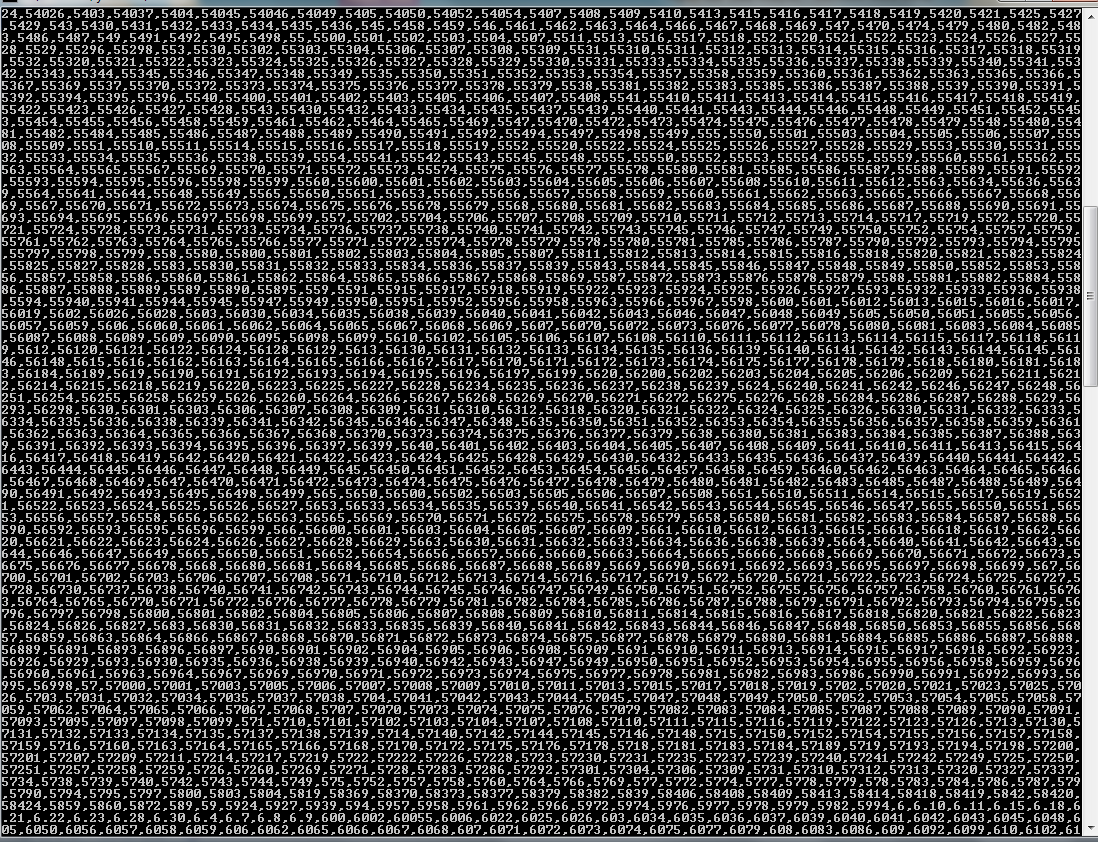
If user selects to see all distinct AS connections, please enter “y”.

Following is the screenshot of “y”request.

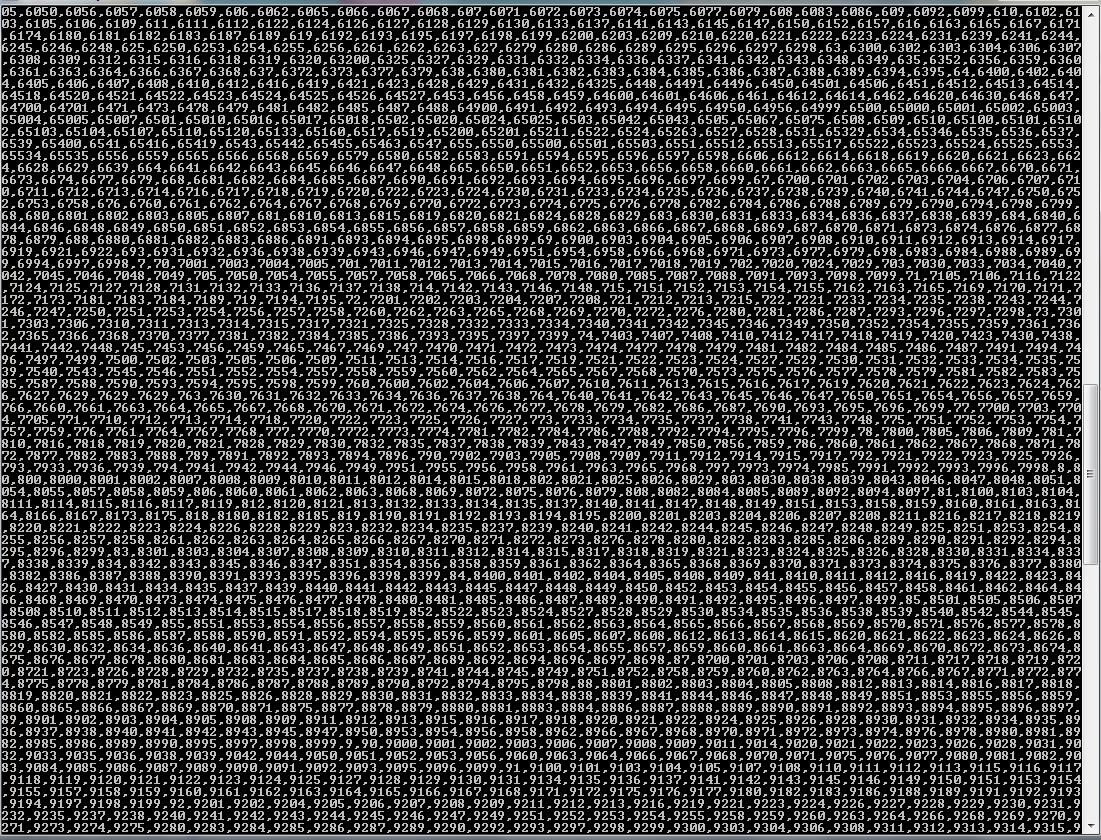
1. First dataset of AS’s:



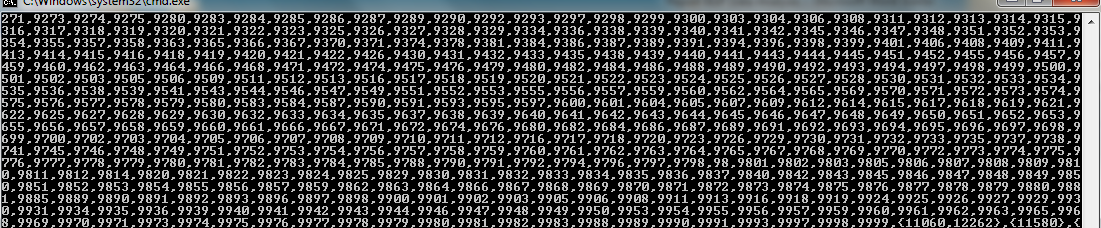
1. 2nd DataSet of Distinct AS’s :



1. 3rd data set of Distinct AS’s



1. Last Data set of As’s



Now the answers :

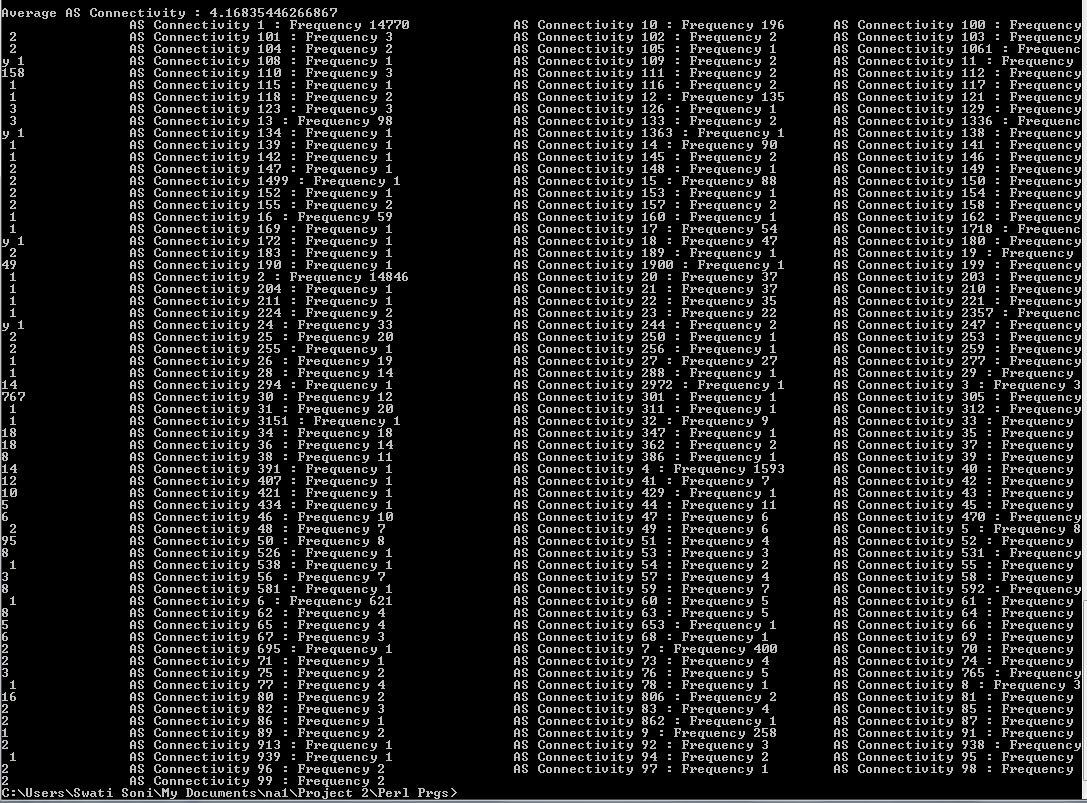
**Answer 2: Distinct AS in the network : 39203**

**Answer 4a: Maximum AS connectivity : 3151**

**Answer 4b: Average AS connectivity : 4.16 (Calculated as :: Total AS Connectivity / Total AS in the network).**



Now for Answer 5 and the graph we have : following connectivity vs frequency data, which is very hard to plot in excel.



**Observation :**

1. It is found that the one having maximum connectivities are rare to be found while those with minimum connectivity are huge in number.
2. Last Sequence number in the Data file corresponds to the total number of prefixes in the BGP data.

The result in the end of the analysis shows a complete Frequency vs. AS-AS connectivity graph, which is found to be very huge. I have provided the proper analysis of connectivity’s vs. frequency of BGP connections.

**Code for Answers 2,3 and 4:**

For questions 2,4 and 5 please consider the following code of data. It mentions about how to compute the Total prefixes and distinct AS connections and further the one-many relationship between AS-AS connections.

#! /usr/bin/perl -w

while(<>){

while(/(string(\d{1,3}))/g){

print "$1\n" if $2 <= 100;

}

}

#FH1 = open "abc.txt";

#open(FH1,"outputRib.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFileAS209\_6.dat")|| die "can not open the file \n";

#@abcfile = <FH1>;

$wordPrefix ="PREFIX";

$wordAS\_Path ="AS\_PATH";

$count=0;

$subnetStr="";

$preSubnet="";

$nextSubnet="";

$distinctPrefixes=0;

%hashASTable ;

#start Time

$start = time();

@FilesToRead = ("ribfinal4920000I.dat","ribfinal4920000II.dat","ribfinal4920000III.dat","ribfinal4920000IV.dat","ribfinal4920000V.dat","ribfinal4920000VI.dat");

foreach my $p (@FilesToRead)

{

open(FH1,$p)|| die "can not open the file \n";

@abcfile = <FH1>;

foreach $line(@abcfile){

#print "Line : $line";

$prefixIndex = index($line,$wordPrefix);

$as\_pathIndex = index($line,$wordAS\_Path);

if($as\_pathIndex>=0){

my @as\_pathValues = split('\s',$line);

my $as\_pathlength = (scalar @as\_pathValues);

for($i=0; $i<$as\_pathlength; $i++){

#print "\nAS Number : $as\_pathValues[$i]\t";

#check if the AS\_Path is in line.

if($wordAS\_Path.":" ne $as\_pathValues[$i]){

if(exists($hashASTable{$as\_pathValues[$i]})){

#print "\nExisting AS Key : $as\_pathValues[$i]\n";

if($i>1){

$j=$i-1;

$preHashKeyVal = $as\_pathValues[$j];

#print "Precceedor AS".$preHashKeyVal;

# add the value of current AS to the pervious AS Links

$IsSameValue = 0;

if($as\_pathValues[$i] ~~ $preHashKeyVal ){

#print "Prev value and as value are equal";

$IsSameValue=1;

}

# add the value of current AS to the pervious AS Link

if($IsSameValue == 0){

$DontADD=0;

foreach $raw\_data(@{$hashASTable{$preHashKeyVal}})

{

if (index($raw\_data,$as\_pathValues[$i]) ge 0)

{

# print "\nFound the AS in @{$hashASTable{$preHashKeyVal}}\n";

$DontADD=1;

}

}

if($as\_pathValues[$i] ~~ @{$hashASTable{$preHashKeyVal}})

{

$DontADD=1;

}

if($DontADD==0){

push @{$hashASTable{$preHashKeyVal}},$as\_pathValues[$i];

# print " Add Value to Key ";

}

#print "\n$preHashKeyVal has Links of @{$hashASTable{$preHashKeyVal}}";

# add previous AS in the current AS links

$DontADD=0;

foreach $raw\_data(@{$hashASTable{$as\_pathValues[$i]}})

{

if (index($raw\_data,$as\_pathValues[$i]) ge 0)

{

# print "\nFound the AS in @{$hashASTable{$preHashKeyVal}}\n";

$DontADD=1;

}

}

if($preHashKeyVal ~~ @{$hashASTable{$as\_pathValues[$i]}})

{

$DontADD=1;

}

if($DontADD==0){

#$hashASTable{$as\_pathValues[$i]} = $preHashKeyVal ;

push @{$hashASTable{$as\_pathValues[$i]}},$preHashKeyVal;

#print " ADD $preHashKeyVal to $as\_pathValues[$i] ";

}

}

}

}else{

if($i==1){

#$hashASTable{$as\_pathValues[$i]}="";

#push @{$hashASTable{$as\_pathValues[$i]}},999999999999;

#print("\nFirst hash table key AS: ",join (',',sort keys %hashASTable),"\n");

}else{

$j=$i-1;

#print "j : $j and i : $i";

$preHashKeyVal = $as\_pathValues[$j];

#print $preHashKeyVal," and ",$as\_pathValues[$i];

$IsSameValue = 0;

if($as\_pathValues[$i] ~~ $preHashKeyVal ){

#print "Prev value and as value are equal";

$IsSameValue=1;

}

# add the value of current AS to the pervious AS Link

if($IsSameValue == 0){

$DontADD=0;

foreach $raw\_data(@{$hashASTable{$preHashKeyVal}})

{

if (index($raw\_data,$as\_pathValues[$i]) ge 0)

{

# print "\nFound the AS in @{$hashASTable{$preHashKeyVal}}\n";

$DontADD=1;

}

}

if($as\_pathValues[$i] ~~ @{$hashASTable{$preHashKeyVal}})

{

$DontADD=1;

}

if($DontADD == 0){

push @{$hashASTable{$preHashKeyVal}},$as\_pathValues[$i];

}

#print "\n$preHashKeyVal has Links of @{$hashASTable{$preHashKeyVal}}";

# add previous AS in the current AS links

push @{$hashASTable{$as\_pathValues[$i]}},$preHashKeyVal;

#print "\n$as\_pathValues[$i] has links of @{$hashASTable{$as\_pathValues[$i]}}"

}

}

# print "Added to the Hash table: $as\_pathValues[$i]\n ";

}

}

}

}

if($prefixIndex>=0){

#print $line;

#($name,$subnet) = split("\s",$line);

# print "Name : $name\t"."Subnet : $subnet";

my @values = split('\s',$line);

# foreach my $val (@values){

# print "$val\n";

# }

($first,$value) = split('\s',$line);

if($count==0){

$preSubnet=$value;

}else{

$nextSubnet=$value;

}

if($preSubnet ne $nextSubnet){

#print "Pre subnet :$preSubnet \t"."next Subnet: $nextSubnet";

$distinctPrefixes++;

$subnetStr=$subnetStr."\n".$value;

if($count>0){

$preSubnet=$nextSubnet;

}

}

$count++;

}

}

close(FH1);

}

# end timer

$end = time();

my @sortHashASTable = sort keys %hashASTable;

print "\nAS Connectivity for All AS\n";

print "AS : <AS LIST> ==>";

$MaximumASConnectivity = 0;

$AverageASConnectivity = 0;

#Declare Hash for Key : X axis : As Connectivity Degree and Y axis : frequency of AS with that Degree

%graph;

foreach $key ( sort keys %hashASTable){

print "$key : @{$hashASTable{$key}} \t";

$ASConnectivity = scalar( @{$hashASTable{$key}} );

# Key of the table is $ASConnectivity

if($ASConnectivity ~~ %graph){

$prevFreq = @{$graph{$ASConnectivity}}[0];

$prevFreq++;

#push @{$hashASTable{$as\_pathValues[$i]}},$preHashKeyVal;

@{$graph{$ASConnectivity}}[0] = $prevFreq;

} else {

push @{$graph{$ASConnectivity}},1;

}

$AverageASConnectivity = $AverageASConnectivity + $ASConnectivity;

#print "\n AverageASConnectivity : $AverageASConnectivity \n";

if($MaximumASConnectivity < $ASConnectivity){

$MaximumASConnectivity = $ASConnectivity;

}

}

print "\nDistinct Subnets: $subnetStr\n";

# report

print "\n\nTime taken was ", ($end - $start), " seconds";

print "\nNumber of Distinct prefixes :$distinctPrefixes";

print "\n Do you wnat to see all distinct AS in this file? (y/n)";

$choice = <>;

chomp $choice;

if("y" ~~ $choice){

print("\nDistinct AS: ",join (',',sort keys %hashASTable),"\n\n");

}

print "\nNumber of Distinct AS in BGP Data:",scalar keys %hashASTable,"";

print "\nMaximum AS Connectivity : $MaximumASConnectivity";

$TotalDistinctAS = scalar keys %hashASTable;

print "\nTotalDistinctAS : $TotalDistinctAS ";

print "\nTotalASConnectivity : $AverageASConnectivity \n";

$AverageASConnectivity = $AverageASConnectivity / $TotalDistinctAS;

print "\nAverage AS Connectivity : $AverageASConnectivity\n ";

foreach $key ( sort keys %graph){

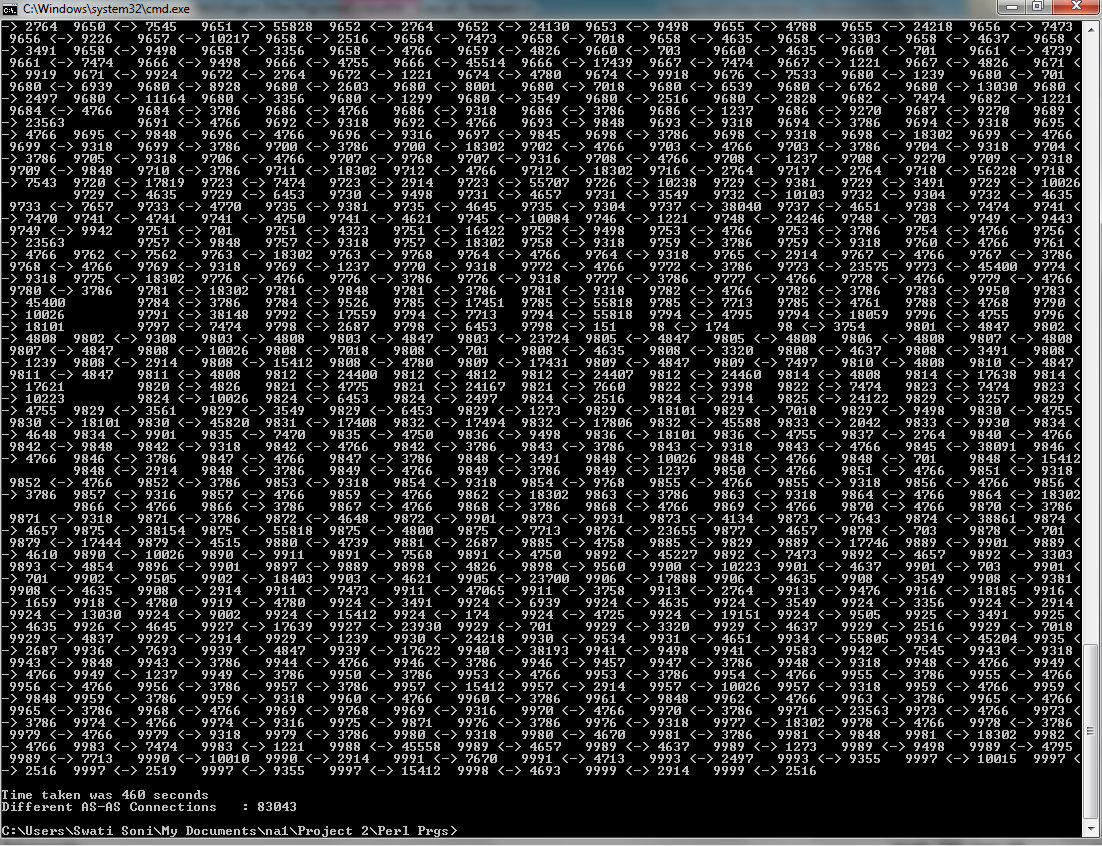
print "\t\tAS Connectivity $key : Frequency @{$graph{$key}}";

}

1. **How many different AS-AS connections can be inferred from the RIBs?**

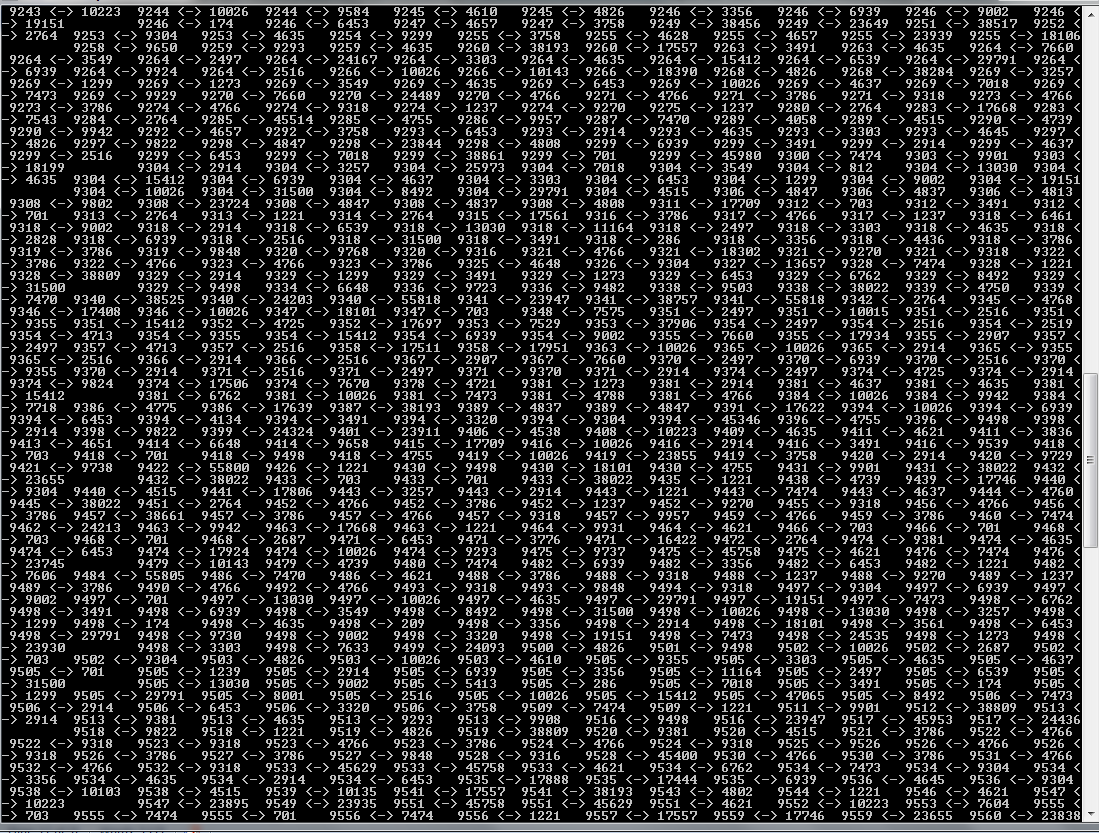
Time Taken for analysis of total BGP data is : 481 seconds

Answer : 83043



Further AS-AS connections can be seen as :







I analyzed only one AS-AS connection amongst all BGP AS connections. Also AS1 connected to AS2 is considered the same as AS2 connected to AS1.

#! /usr/bin/perl -w

while(<>){

while(/(string(\d{1,3}))/g){

print "$1\n" if $2 <= 100;

}

}

#FH1 = open "abc.txt";

#open(FH1,"outputRib.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFileAS209\_6.dat")|| die "can not open the file \n";

#@abcfile = <FH1>;

$wordPrefix ="PREFIX";

$wordAS\_Path ="AS\_PATH";

$count=0;

$subnetStr="";

$preSubnet="";

$nextSubnet="";

$distinctPrefixes=0;

%hashASTable ;

#start Time

$start = time();

#@FilesToRead = ("ribfinal4920000I.dat");

#ribfinal4920000I.dat");

#,"ribfinal4920000II.dat","ribfinal4920000III.dat","ribfinal4920000IV.dat",,"ribfinal4920000V.dat",,"ribfinal4920000VI.dat");

@FilesToRead = ("ribfinal4920000I.dat","ribfinal4920000II.dat","ribfinal4920000III.dat","ribfinal4920000IV.dat","ribfinal4920000V.dat","ribfinal4920000VI.dat");

foreach my $p (@FilesToRead)

{

open(FH1,$p)|| die "can not open the file \n";

@abcfile = <FH1>;

foreach $line(@abcfile){

#print "Line : $line";

$prefixIndex = index($line,$wordPrefix);

$as\_pathIndex = index($line,$wordAS\_Path);

if($as\_pathIndex>=0){

my @as\_pathValues = split('\s',$line);

my $as\_pathlength = (scalar @as\_pathValues);

for($i=0; $i<$as\_pathlength; $i++){

#print "\nAS Number : $as\_pathValues[$i]\t";

#check if the AS\_Path is in line.

if($wordAS\_Path.":" ne $as\_pathValues[$i]){

if(exists($hashASTable{$as\_pathValues[$i]})){

#print "\nExisting AS Key : $as\_pathValues[$i]\n";

if($i>1){

$j=$i-1;

$preHashKeyVal = $as\_pathValues[$j];

#print "Precceedor AS".$preHashKeyVal;

# add the value of current AS to the pervious AS Links

$IsSameValue = 0;

if($preHashKeyVal==$as\_pathValues[$i]){

#print "Prev value and as value are equal";

$IsSameValue=1;

}

# add the value of current AS to the pervious AS Link

if($IsSameValue == 0){

#print "\n$preHashKeyVal has Links of @{$hashASTable{$preHashKeyVal}}";

# add previous AS in the current AS links

$DontADD=0;

foreach $raw\_data(@{$hashASTable{$as\_pathValues[$i]}})

{

if (index($raw\_data,$as\_pathValues[$i]) ge 0)

{

# print "\nFound the AS in @{$hashASTable{$preHashKeyVal}}\n";

$DontADD=1;

}

}

if($preHashKeyVal ~~ @{$hashASTable{$as\_pathValues[$i]}})

{

$DontADD=1;

}

if($DontADD==0){

#$hashASTable{$as\_pathValues[$i]} = $preHashKeyVal ;

push @{$hashASTable{$as\_pathValues[$i]}},$preHashKeyVal;

#print " ADD $preHashKeyVal to $as\_pathValues[$i] ";

}

}

}

}else{

if($i==1){

#print("\nFirst hash table key AS: ",join (',',sort keys %hashASTable),"\n");

}else{

$j=$i-1;

#print "j : $j and i : $i";

$preHashKeyVal = $as\_pathValues[$j];

#print $preHashKeyVal," and ",$as\_pathValues[$i];

$IsSameValue = 0;

if($preHashKeyVal==$as\_pathValues[$i]){

#print "Prev value and as value are equal";

$IsSameValue=1;

}

# add the value of current AS to the pervious AS Link

if($IsSameValue == 0){

#print "\n$preHashKeyVal has Links of @{$hashASTable{$preHashKeyVal}}";

# add previous AS in the current AS links

push @{$hashASTable{$as\_pathValues[$i]}},$preHashKeyVal;

}

}

# print "Added to the Hash table: $as\_pathValues[$i]\n ";

}

}

}

}

}

close(FH1);

}

# end timer

$end = time();

my @sortHashASTable = sort keys %hashASTable;

print "\nAS Connectivity for All AS\n";

print "AS <-> AS \n";

foreach $key ( sort keys %hashASTable){

$ASConnectivity = scalar( @{$hashASTable{$key}} );

if($ASConnectivity > 0){

foreach (@{$hashASTable{$key}}) {

print "\t $key <-> ".$\_;

}

}

$AverageASConnectivity = $AverageASConnectivity + $ASConnectivity;

#print "\n AverageASConnectivity : $AverageASConnectivity \n";

}

# report

print "\n\nTime taken was ", ($end - $start), " seconds";

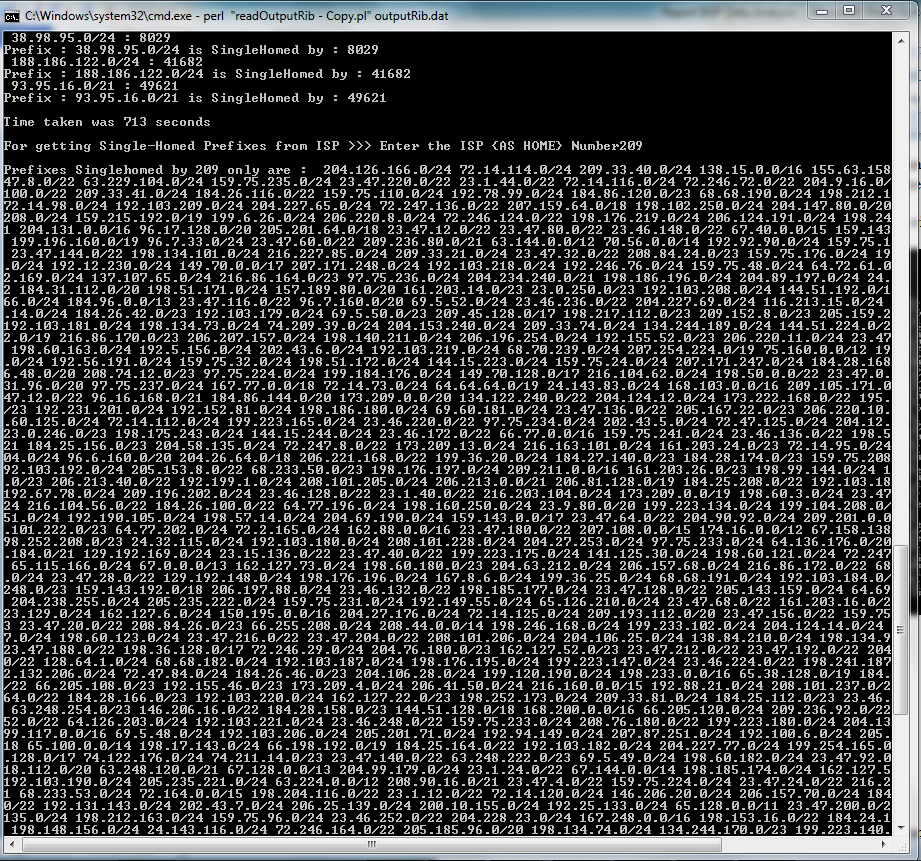
print "\nDifferent AS-AS Connections : $AverageASConnectivity \n";

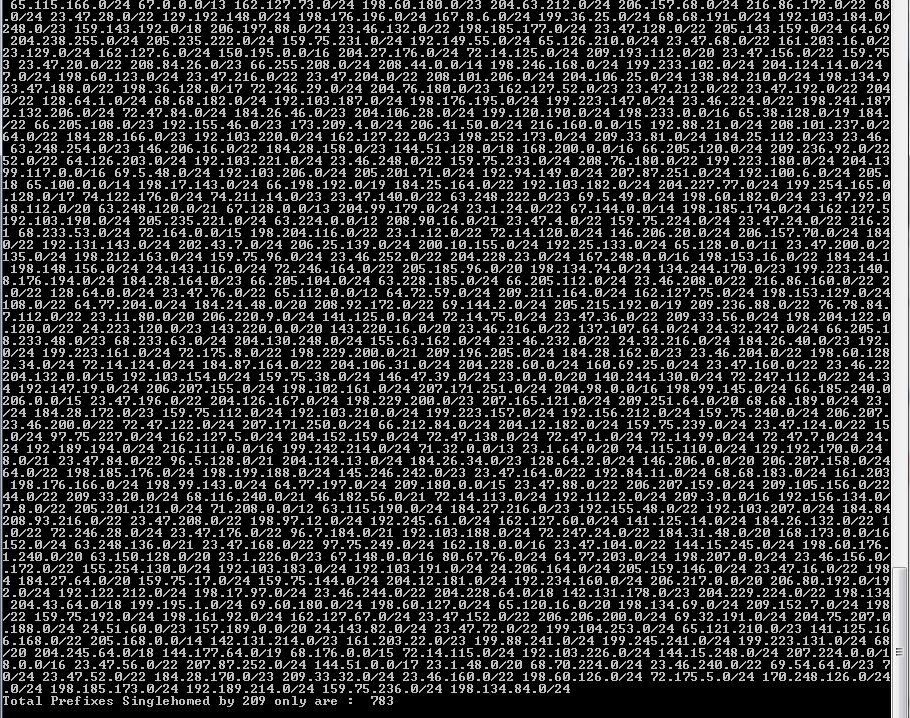
**Extra- Credit Questions : On Multi-homing and Single-homing:**

1. **How many Prefixes are originated from ISP-A only (i.e. single-homed Prefixes from ISP-A )?**

**For the analysis of this particular problem**

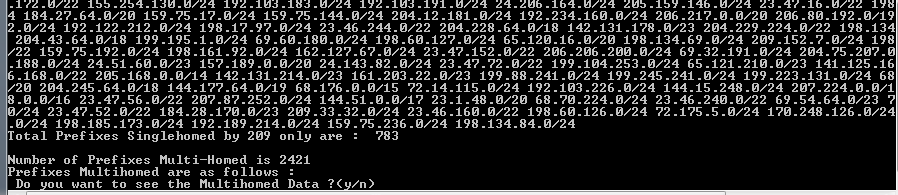
* I analyzed all the prefixes that belong to an AS.
* A prefix is homed by the “last AS” in AS\_PATH field of the BGP data.
* So in this particular question I collected all prefixes along with their corresponding homes into a hash map with Key as prefixes and AS values as homes.
* Now this hash map is analyzed and a singlehomedHash and MultiHomedHash maps as constructed.
* For determining prefixes of a particular AS, user is asked for AS number of his choice.
* Suppose user entered: 209.
* Now after receiving AS from user entire hash table is searched for the AS number.
* The prefixes (Keys) which are associated to only one AS number and that one is “<USER\_AS\_NUMBER>” are collected in another hash map.
* The number of Prefixes single homed by “<USER\_AS\_NUMBER>” along with the values of prefixes are shown to the user.



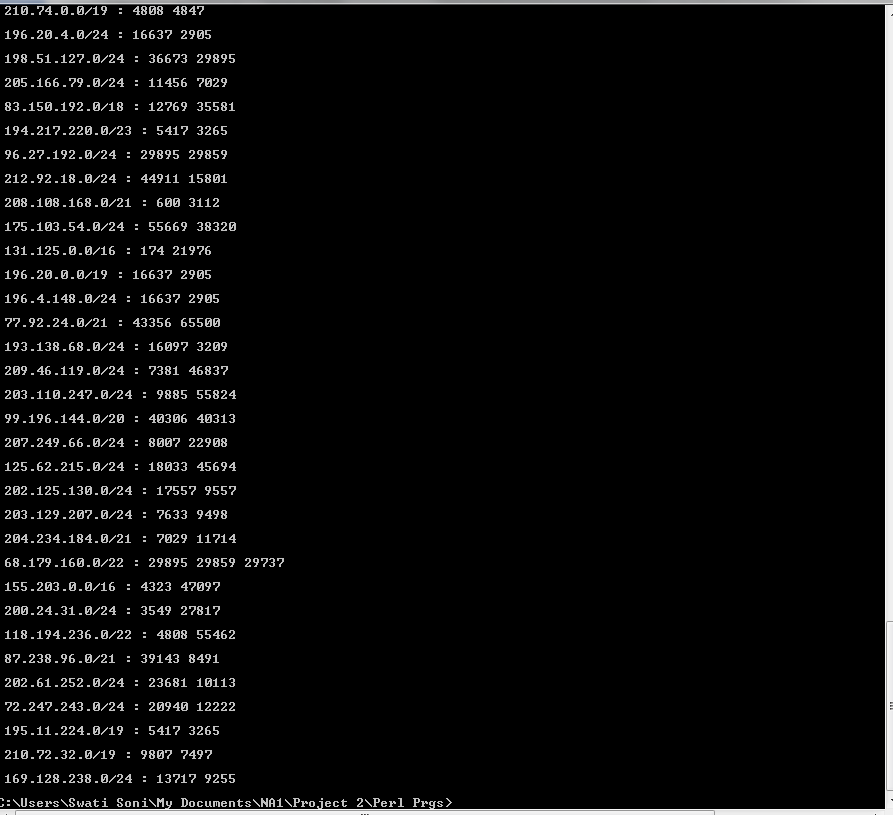


**Prefixes multiple-homed by 209 is 783.**

1. **How many prefixes are multi-homed?**
2. The logic for framing the AS maps is mentioned previously in answer 6
3. Following is the screen shot for the analyzed multi-homed data :



1. So the entire Multi-homed prefixes are 2421.
2. I also provided the option for viewing the multi-homed data. So when the user can provide his choice for viewing the BGP data by either providing y or n when asked by program.
3. On ‘y’ following list of multi-homed Prefixes if displayed :



**Following is the Perl script for Q6 and Q7 :**

#! /usr/bin/Perl -w

while(<>){

while(/(string(\d{1,3}))/g){

print "$1\n" if $2 <= 100;

}

}

#FH1 = open "abc.txt";

@FilesToRead = ("ribfinal4920000I.dat","ribfinal4920000II.dat","ribfinal4920000III.dat","ribfinal4920000IV.dat",,"ribfinal4920000V.dat",,"ribfinal4920000VI.dat");

#open(FH1,"outputRib.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFileAS209\_6.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile100000lines.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile640000lines.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile1280000lines.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile2460000lines.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile4920000lines.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile9840000lines.dat")|| die "can not open the file \n";

#open(FH1,"RibAsciiFile19680000lines.dat")|| die "can not open the file \n";

$wordPrefix ="PREFIX";

$wordAS\_Path ="AS\_PATH";

$count=0;

$subnetStr="";

$preSubnet="";

$nextSubnet="";

$distinctPrefixes=0;

%hashASTable ;

$PrefixVal;

$prefixHome;

%hashPrefixHome ;

$pushAt2 = 0;

$start = time();

foreach my $p (@FilesToRead)

{

open(FH1,$p)|| die "can not open the file \n";

@abcfile = <FH1>;

foreach $line(@abcfile){

#print "Line : $line";

$prefixIndex = index($line,$wordPrefix);

$as\_pathIndex = index($line,$wordAS\_Path);

if($pushAt2 ==2){

$pushAt2 = 0;

}

# add prefixes as keys with homes as last element of AS\_PATH.

if($prefixIndex>=0){

#print $line;

#($name,$subnet) = split("\s",$line);

my @values = split('\s',$line);

# foreach my $val (@values){

# print "$val\n";

# }

($first,$value) = split('\s',$line);

if($count==0){

$preSubnet=$value;

}else{

$nextSubnet=$value;

}

$pushAt2++;

if($preSubnet ne $nextSubnet){

#print "Pre subnet :$preSubnet \t"."next Subnet: $nextSubnet";

$distinctPrefixes++;

$subnetStr=$subnetStr."\n".$value;

#add the Distinct Prefix into the Key into hash Table.

$PrefixVal = $value;

#print "\n PrefixVal : $PrefixVal At $pushAt2";

if($count>0){

$preSubnet=$nextSubnet;

}

}

$count++;

}

if($as\_pathIndex>=0){

my @as\_pathValues = split('\s',$line);

my $as\_pathlength = (scalar @as\_pathValues);

$as\_pathlength--;

$pushAt2++;

$prefixHome = $as\_pathValues[$as\_pathlength];

#print " prefixHome : $prefixHome At $pushAt2";

}

#print "pushAt2 : $pushAt2";

if($pushAt2 == 2) {

# perform push into hash with this.

#print "\n\n PrefixVal : $PrefixVal At $pushAt2";

#print "\n prefixHome : $prefixHome At $pushAt2";

if( $prefixHome ~~ @{$hashPrefixHome{$PrefixVal}}) { }

else{

push @{$hashPrefixHome{$PrefixVal}},$prefixHome;

}

}

}

close(FH1);

}

#file processing closes here

$prefixesMultiHomed = 0;

%SingleHomedPrefixes;

%MultiHomedPrefixes;

print "\n Prefix : Homes ";

foreach $key ( keys %hashPrefixHome){

print "\n $key : @{$hashPrefixHome{$key}} \n";

$homesForPrefix = scalar @{$hashPrefixHome{$key}};

if( $homesForPrefix > 1 ){

$prefixesMultiHomed++;

print "Prefix : $key is Multihomed by : @{$hashPrefixHome{$key}}";

push @{$MultiHomedPrefixes{$key}},@{$hashPrefixHome{$key}};

}else {

print "Prefix : $key is SingleHomed by : @{$hashPrefixHome{$key}}";

push @{$SingleHomedPrefixes{@{$hashPrefixHome{$key}}[0]}},$key;

}

}

# end timer

$end = time();

# report

print "\n\nTime taken was ", ($end - $start), " seconds";

print "\n\nFor getting Single-Homed Prefixes from ISP >>> Enter the ISP {AS HOME} Number";

$AS\_Numb = <>;

chomp $AS\_Numb ;

print "\nPrefixes Singlehomed by $AS\_Numb only are : @{$SingleHomedPrefixes{$AS\_Numb}}";

$numOfPrefixes = scalar @{$SingleHomedPrefixes{$AS\_Numb}};

print "\nTotal Prefixes Singlehomed by $AS\_Numb only are : $numOfPrefixes ";

print "\n\nNumber of Prefixes Multi-Homed is $prefixesMultiHomed";

print "\nPrefixes Multihomed are as follows :";

print "\n Do you want to see the Multihomed Data ?(y/n)";

$choice = <>;

chomp $choice;

if("y" ~~ $choice){

foreach $key ( keys %MultiHomedPrefixes){

print "\n $key : @{$MultiHomedPrefixes{$key}} \n";

}

}

**READ ME file(file name,usage,functionality)**

Steps to follow for analysis of BGP data are as follows:

1. Download Files from Oregon’s website : <ftp://archive.routeviews.org/route-views.saopaulo/bgpdata/2011.10/RIBS/>
2. Convert the Ribs files downloaded into Ascii Format following the steps:
3. Download zebra-dump-parser.tgz from http://www.linux.if/~md/software/zebra-dump-parser.tgz
4. Un-compress Both the above files using following two commands

tar -xzvf zebra-dump-parser.tgz

bzip2 -d rib.20111001.0000.bz2

1. Now in zebra-dump-parser.pl file change $format = 3 to $format = 1 to obtain the data in correct format.
2. cat rib.20111001.0000 | time ./zebra-dump-parser.pl > ribascii
3. Now ribascii is your ascii file on which you need to perform the data analysis.
4. For further details look into screenshots in Report.
5. Reduce the “ascii” File Contents by running following shell script in Linux.

echo "enter file "

read fname

echo "Extracting Out Prefixes and AS\_Path"

sed '/TIME:/ d' $1 > ripTime.dat

sed '/TYPE:/ d' ripTime.dat > ripTimeType.dat

sed '/FROM:/ d' ripTimeType.dat > ripTimeTypeFrom.dat

sed '/AGGREGATOR:/ d' ripTimeTypeFrom.dat > ripTimeTypeFrom1.dat

sed '/ORIGIN:/ d' ripTimeTypeFrom1.dat > ripTimeTypeFrom11.dat

sed '/ATOMIC\_AGGREGATE/ d' ripTimeTypeFrom11.dat > ripTimeTypeFrom2.dat

sed '/NEXT\_HOP:/ d' ripTimeTypeFrom2.dat > ripTimeTypeFromNexthop.dat

sed '/MULTI\_EXIT\_DISC:/ d' ripTimeTypeFromNexthop.dat > ripTimeTypeFrmNxthpMult.dat

sed '/ORIGINATED:/ d' ripTimeTypeFrmNxthpMult.dat > ripTimeTypeFrmNxthpMultOrg.dat

sed '/COMMUNITIES:/ d' ripTimeTypeFrmNxthpMultOrg.dat > ripTimeTypeFrmNxthpMultOrgCom.dat

sed '/SEQUENCE:/ d' ripTimeTypeFrmNxthpMultOrgCom.dat > ripTimeTypeFrmNxthpMultOrgComSq.dat

sed '/-/ d' ripTimeTypeFrmNxthpMultOrgComSq.dat > ripTimeTypeFrmNxthpMultOrgComSqDs.dat

sed '/^$/ d' ripTimeTypeFrmNxthpMultOrgComSqDs.dat > ripTimeTypeFrmNxthpMultOrgComSqDsSpc.dat

sed '/PEER/ d' ripTimeTypeFrmNxthpMultOrgComSqDsSpc.dat > finalrip.dat

sed '/^$/ d' finalrip.dat > $2

ls

This script is included in file “scriptChopfile” included in the submission of “Programs and ReadME” folder.

1. Each File has its Question number at its end which indicates the question for which the script provides a solution.
2. For proper analysis, Obtain the optimized BGP ascii file from above 3 steps.
3. Chop the files into 4920000 lines by using ,$head option in Linux, for optimized performance of the scripts.
4. Name each block as ("ribfinal4920000I.dat","ribfinal4920000II.dat","ribfinal4920000III.dat","ribfinal4920000IV.dat",,"ribfinal4920000V.dat",,"ribfinal4920000VI.dat") for input for all the scripts.
5. Place all scripts in the same folder as the one containing the chopped files and run the simultaneous scripts.
6. File names , Usage and functionalities:

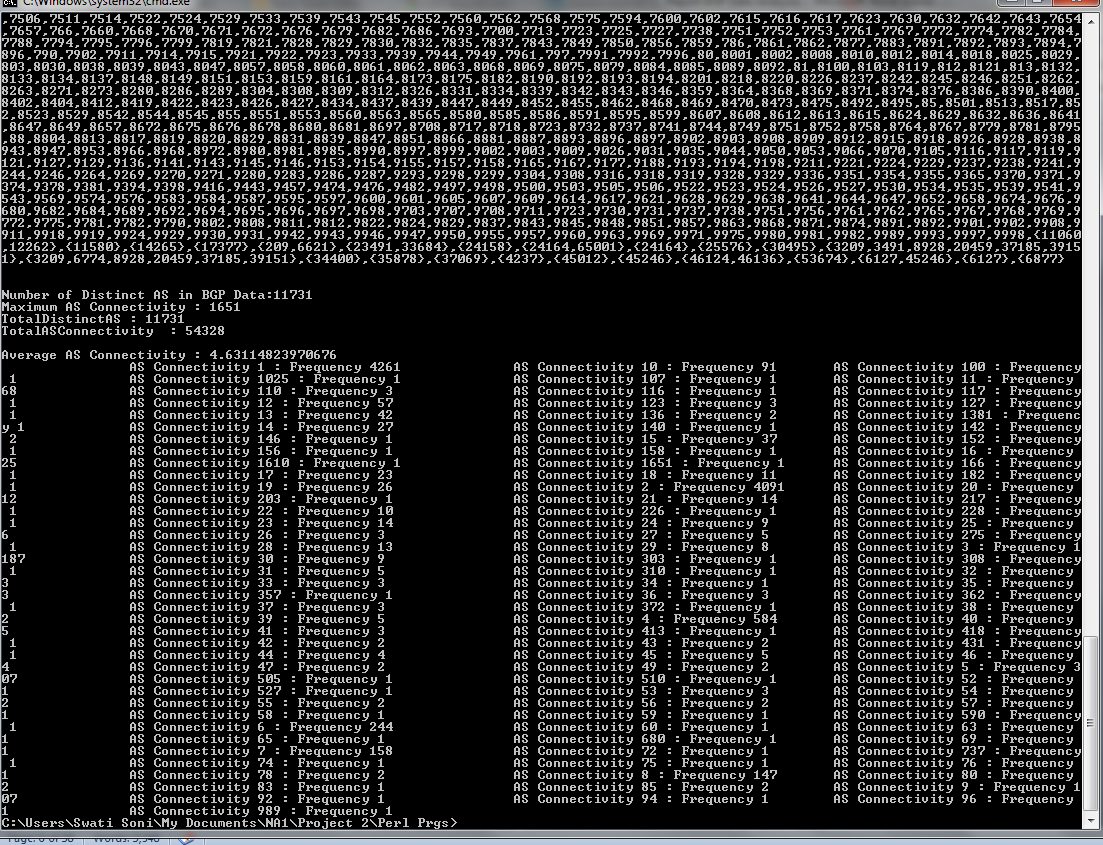
|  |  |  |
| --- | --- | --- |
| File names | Usage | Functionalities |
| readOutputRib\_V1\_Q1\_Q2\_Q4\_Q5 | File for analysis of BGP data for answering Q1,2,4 and 5 respectively | Analysis of BGP data for total prefixes, distinct AS numbers and maps Connectivity of different AS to Frequency in BGP data. |
| readOutputRib\_V2\_Q3 | File for analysis of BGP data for answering Q3 | Analyzes the data and provides one-one AS-AS connections list |
| readOutputRib\_V2\_Q6\_Q7 | File for analysis of BGP data for answering Q6 and Q7 | Maps Prefixes with their homes.  Provides result for multi-home and single-home |
| scriptChopfile | For optimization of BGP input file | Removes unnecessary data and redundancy by eliminating data from files. |
| readOutputRib\_V2\_Q1 | Additional file for same Q1 analysis | More optimized solution. |

**4. Performance Evaluation**

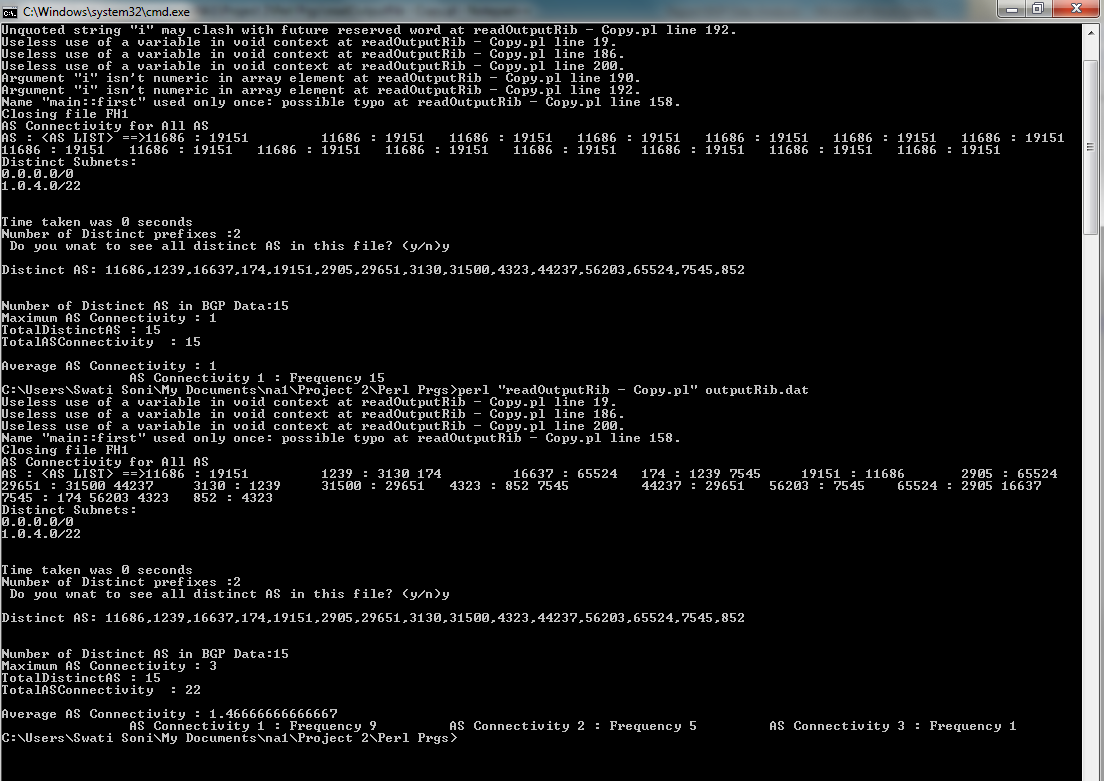
**Approaches-**

1. **Pre-processing of Data :**
2. First I analyzed the BGP sample data. But it was too short to get a complete picture of what’s happening behind the scenes so for proper understanding of the data I converted file from the given link into ascii file.
3. Since these files are very large I used external tool for viewing the large file : large text file viewer.
4. I analyzed the data for the converted ascii file and used it for answering out questions.

First I analysed the BGP data of first chopped file of 4920000 lines for the first 4 questions. Please refer to the screen shot below. This provides an overall view of collective data analysis performed over the BGP data.



For the given sample output file we got the following results:



1. For predicting trends from different times and generations I simultaneously converted the BGP data from previous month of BGP data analysis.
2. After proper analysis, I started working on writing Perl scripts for chopped data’s files using Linux system.
3. **Difficulties faced & Solution**

a. Conversion of Binary to Ascii file : We followed the following steps for conversion of binary file to ascii file :

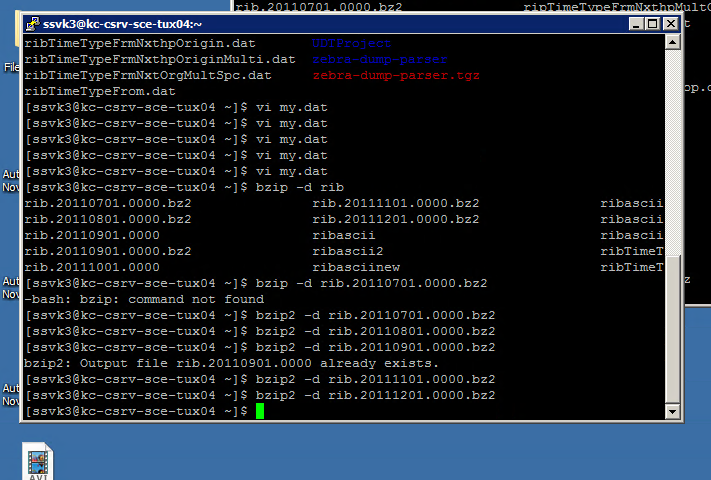
1. Download zebra-dump-parser.tgz from http://www.linux.if/~md/software/zebra-dump-parser.tgz
2. Un-compress Both the above files using following two commands

tar -xzvf zebra-dump-parser.tgz

bzip2 -d rib.20111001.0000.bz2

1. Now in zebra-dump-parser.pl file change $format = 3 to $format = 1 to obtain the data in correct format.
2. cat rib.20111001.0000 | time ./zebra-dump-parser.pl > ribascii

Now ribascii is your ascii file on which you need to perfo.rm the data analysis.



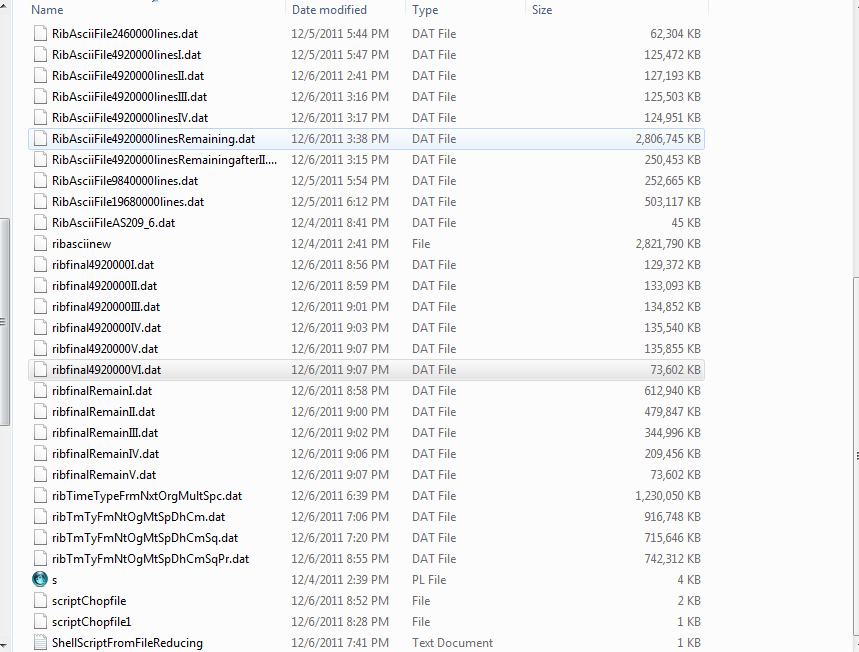
b. These are huge files of 2+GB data so we had to search for tools on Linux and Windows to view the data.

c. Writing algorithms for proper analysis was the next big challenge.

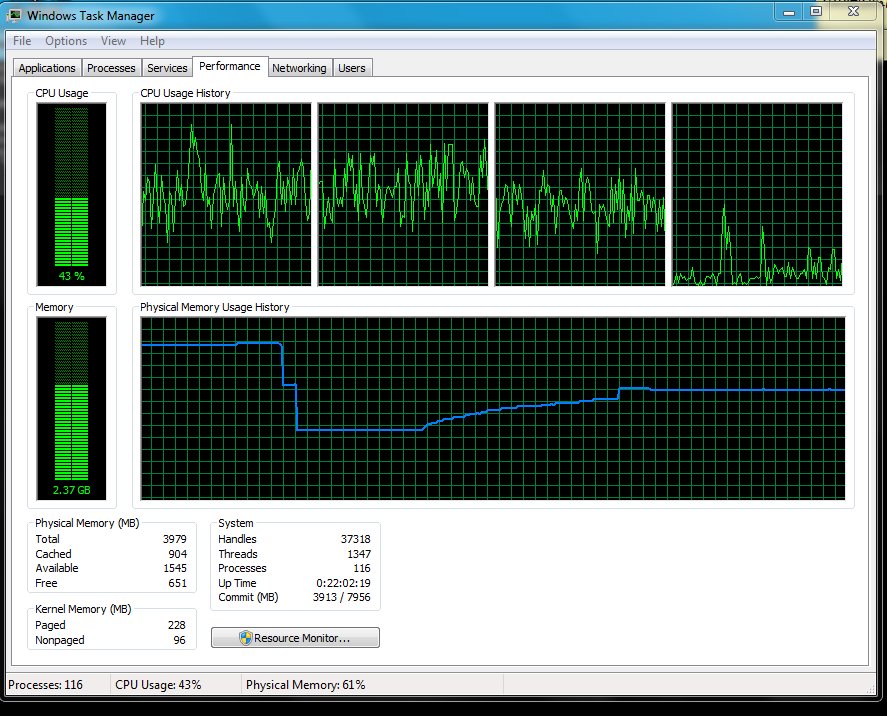
d. Scripting tool selection : I learned Python and Perl for doing scripting . Though I find Perl more useful for the purpose of Data mining from Larger files. It took in 1 week to learn the entire Perl language and play with the data using it over Linux platform

e. Loading huge data : since the memory is limited and I needed to maintain the performance of the system on which we provide results, I used tricks and techniques for analysis of data as mentioned in next section.

Following is the screen shot of files with various sizes I played with:



f. Performance monitoring of the system. Consider the following screenshot where two files of 4920000GBs are loaded one by one into memory. After loading their data’s are analyzed and properly configured.



After carefully viewing the trend in memory utilization and performance of the system, I decided to shop the huge files into 125.5MB’s each for better performance and results.

1. **Techniques/tricks used :**
2. **Trick for handling large volumes of data.**

I faced a huge deal of problem on reading the huge volume of data from BGP. When I first started analyzing BGP data files, I found that with increasing size of input file, the system’s memory utilization increases(an obvious fact if you are loading the entire file in one go) , the system’s performance decreases (laptop freezes) and, so the overall effect was an exponential increase in time taken to evaluated BGP data.

Please consider the following table of my analysis:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Snow. | Lines of BGP Ascii File | Size | System Memory | Time |
| 1. | 64000 | 16 MB | 1.5GB+Few MB | Few Seconds |
| 2 | 1280000 | 32 MB | 1.46 GB+few MB | 20 Seconds |
| 3 | 2460000 | 64 MB | 1.78 GB+ few MB | 45 Seconds |
| 4 | 4920000 | 125.5 MB | 2.2 GB | 75 Seconds |
| 5 | 9840000 | 250 MB | 2.79 GB | 5 min 30 Seconds |
| 6 | 19680000 | 500 MB | 3.98 GB | 1 hr. |

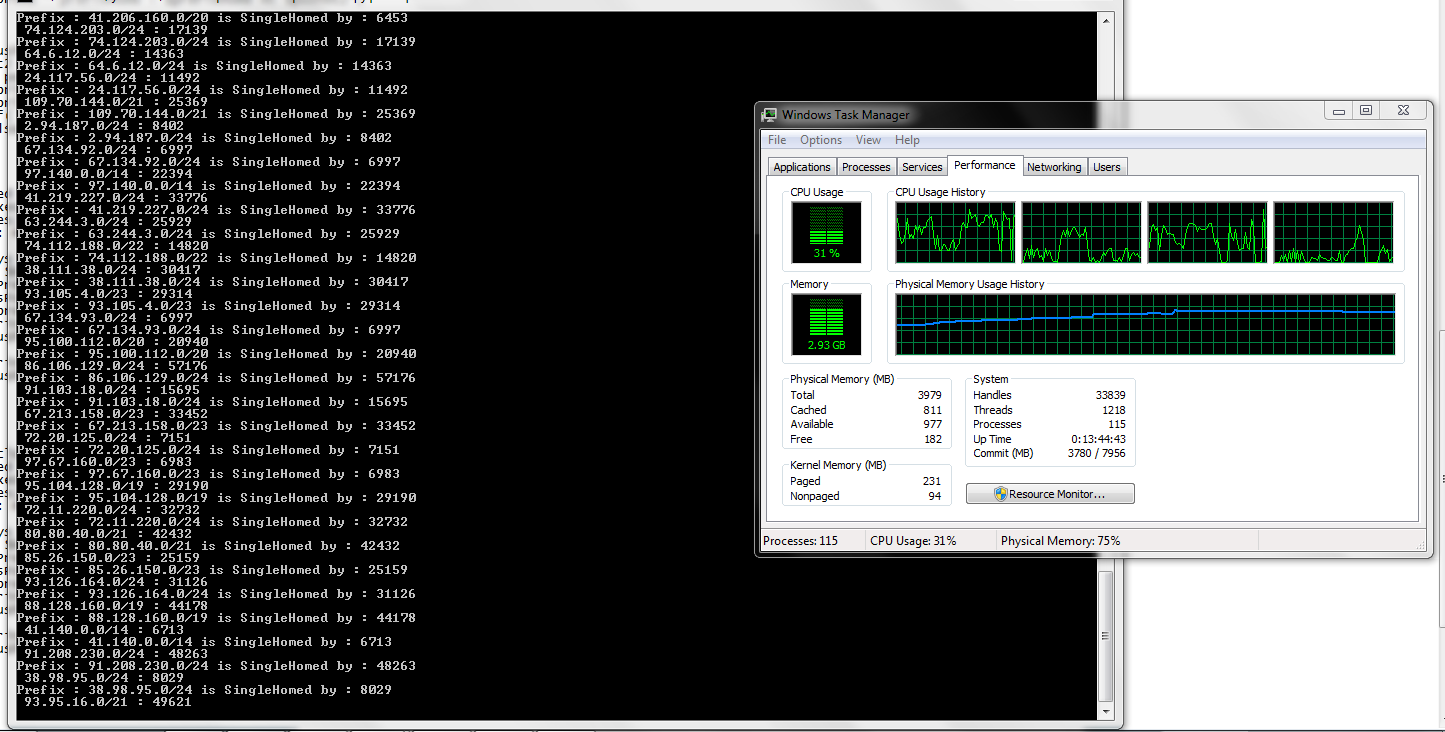
Observing , the impact on memory usage, system’s performance and Time taken for data analysis, I decided to chop the entire file into multiple smaller files.

As opening and analyzing the file for 125.5MB took me around 75 sec, so I decided to chop the file into lines of 4920000 and feed them to the Perl Scripts for proper quick analysis.

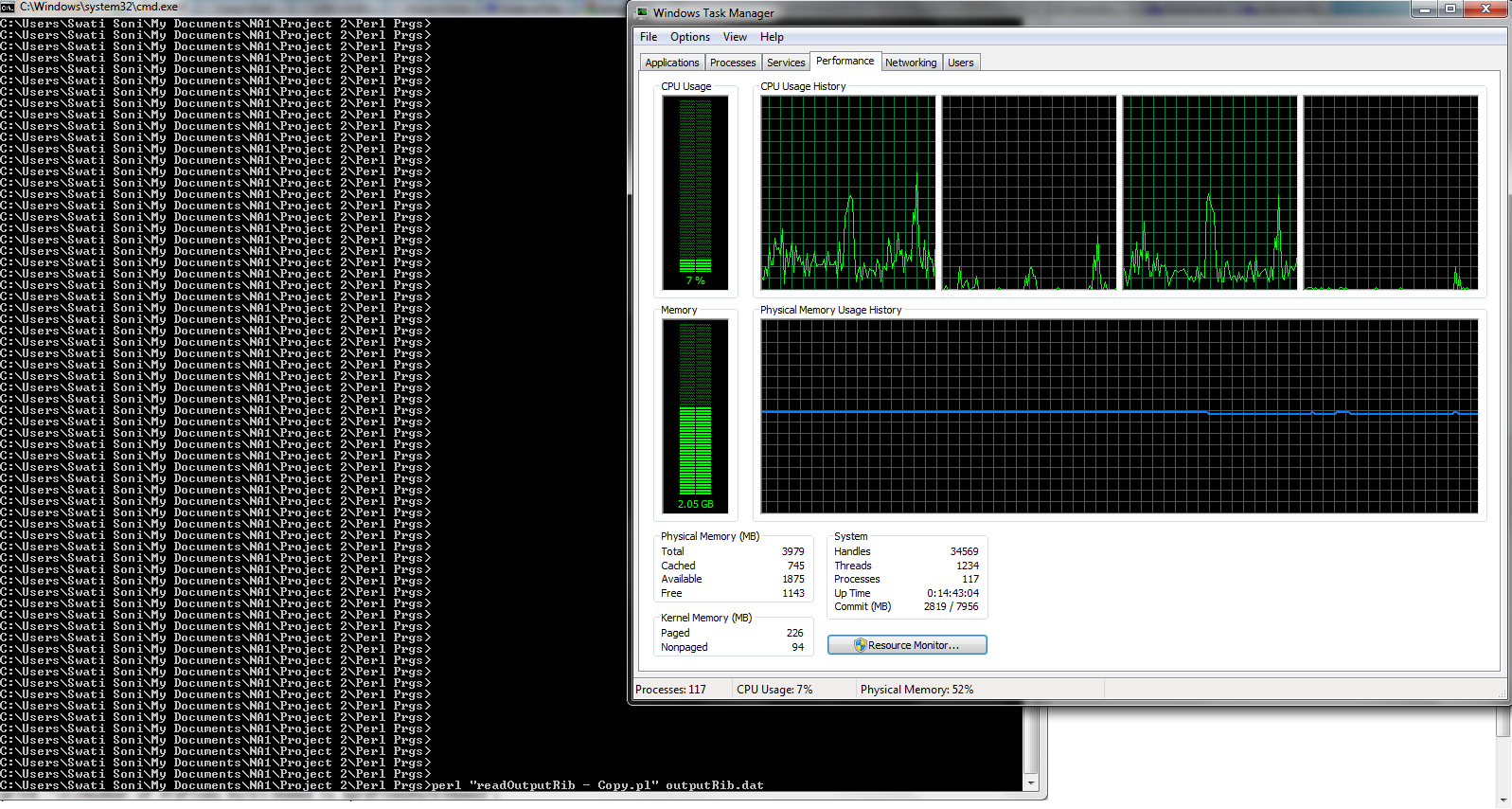
The results were dramatically improved performance of system for reading huge BGP data’s.

For 1 GB files the results were shown in 21 minutes.

The entire file is chopped into 6 files for our analysis . Following is the screen shot for the data after 2nd Chopped file is read :

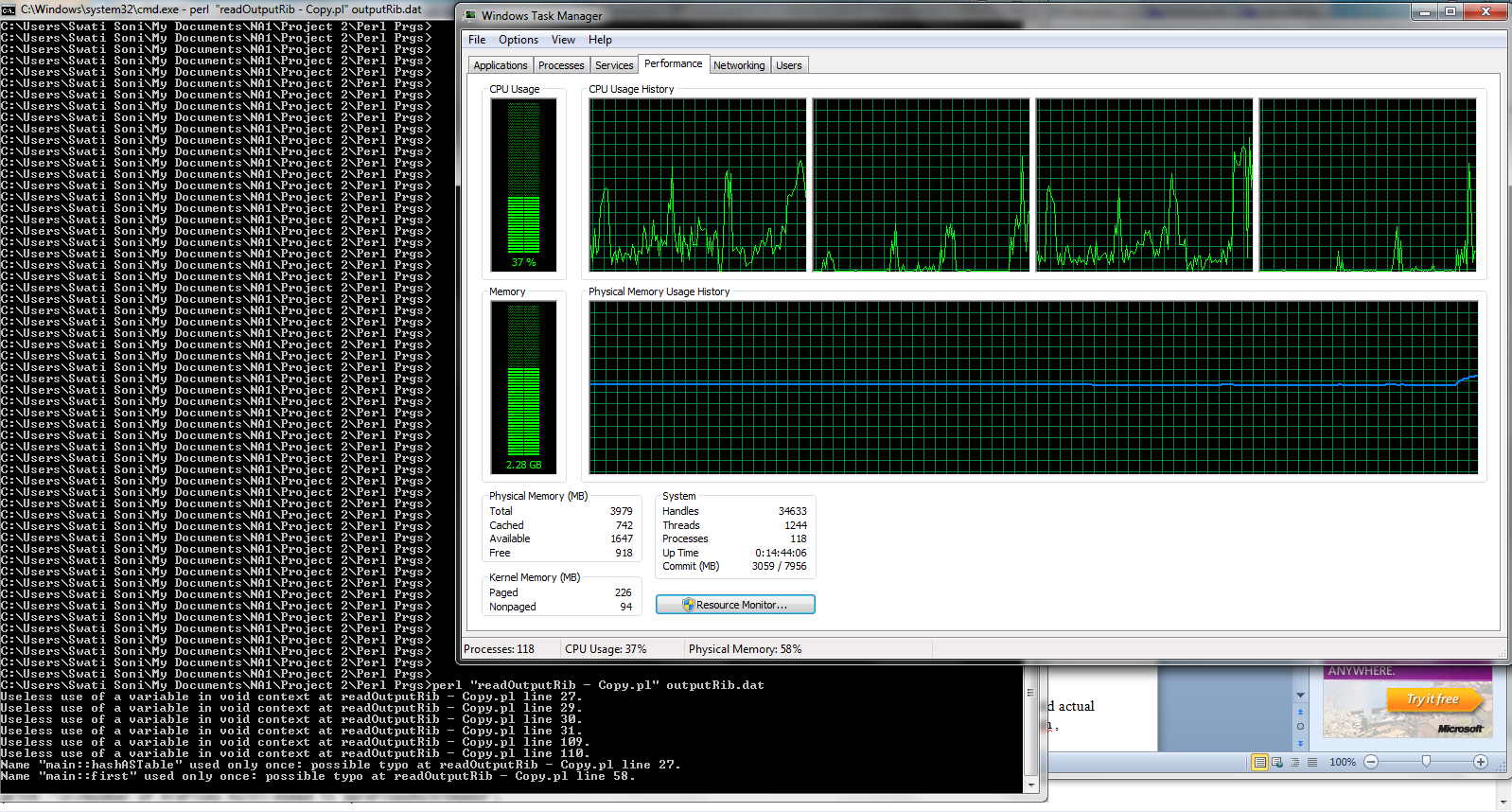


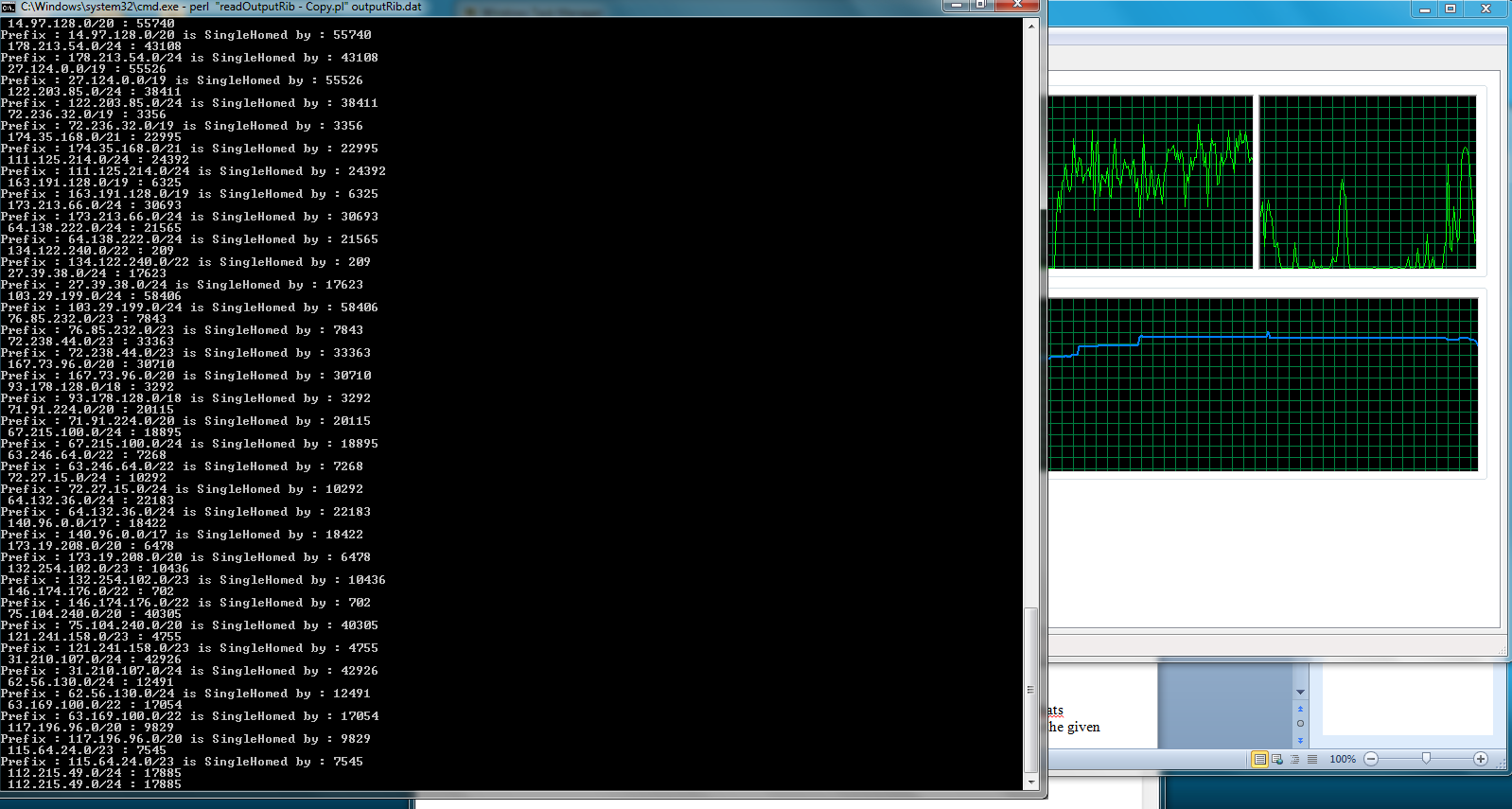
Consider the following screenshot right before the analysis is to begin for Q6 and Q7 simultaneously:



Notice that, my system has already had 2.6GB of memory consumed by various other applications.

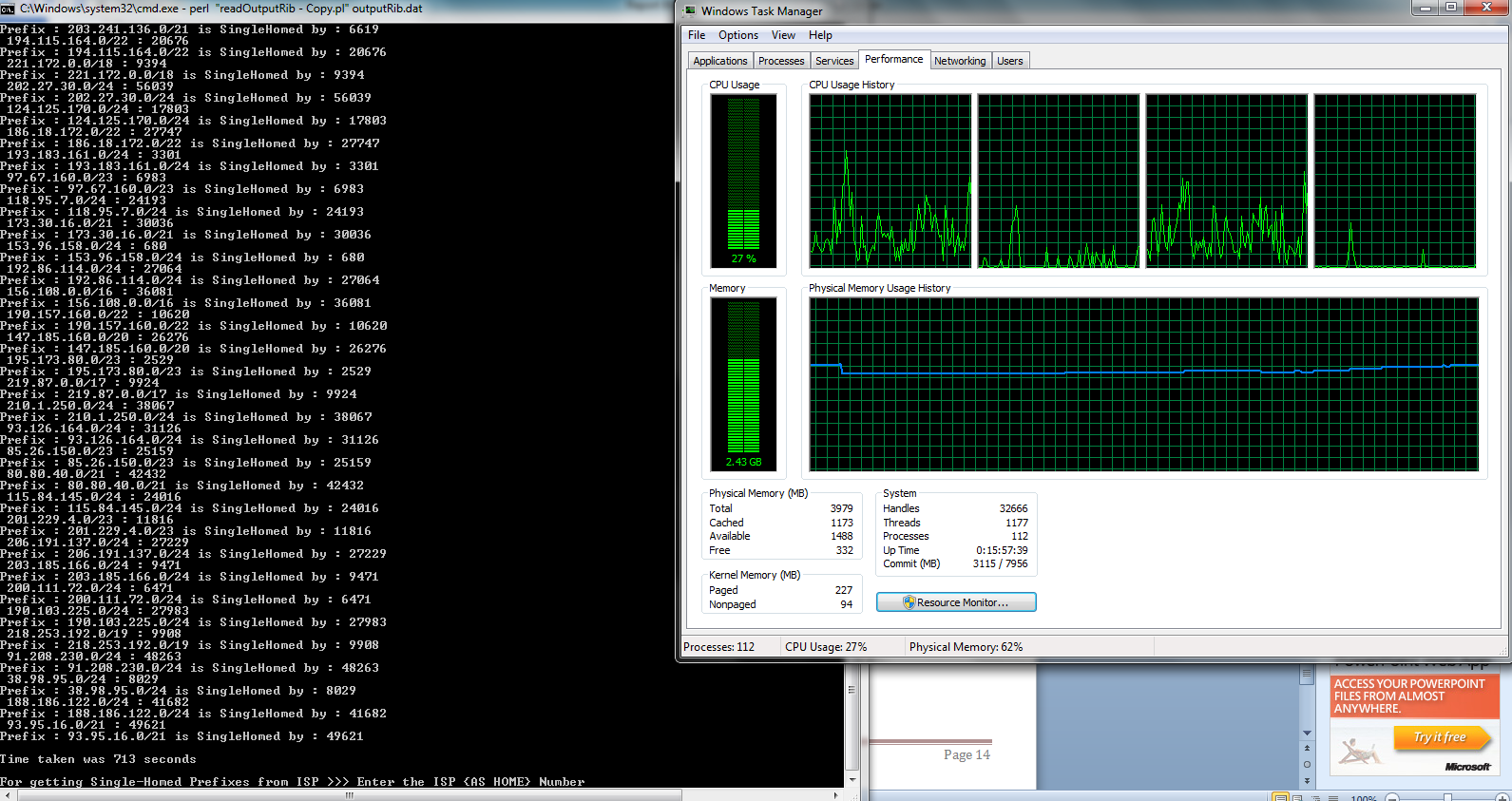
Now I called for the script to run :





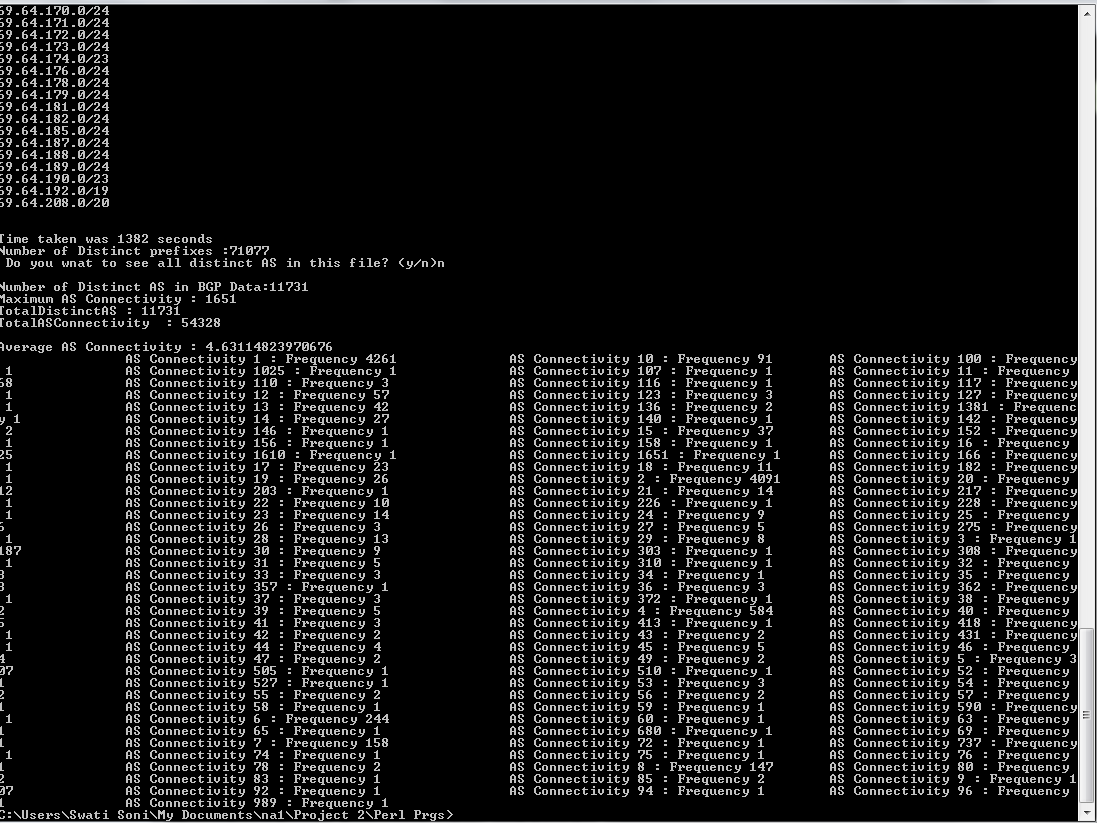
Now consider the system performance, time taken for analysis and memory utilization of my system, from the following screen shot. It’s very much improved.

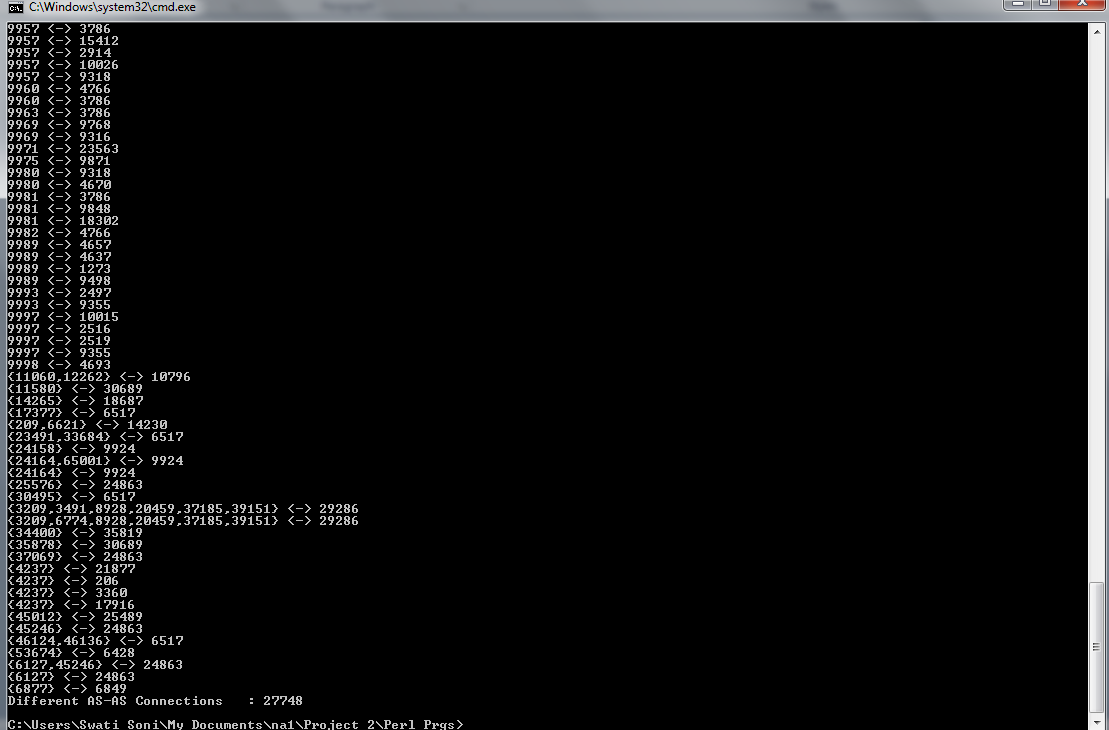
Notice that the time taken now is only “713” seconds ~~ 11 minutes.



1. Consider Questions 1,2,4 and 5 . This requires all the data be properly placed into queues indexed or keyed with prefixes or and other AS’s containing them.

For a 4920000 lines of BGP Ascii data the time taken was 1382seconds for proper analysis of the PGP data.



Also the Different AS-AS connections for the same amount of data is found to be : 

1. **Comments:**
2. BGP data analysis provided me with the opportunity to not only see and work on real world actual routing data but also to think over the various trends the data followed in terms of AS Path , prefixes, Routes taken etc.
3. Support from Our Professor Dr Choi and TA Sunny, for making us understand the real concepts and analogies, was an important factor in our successful completion of the project.
4. **Language/tools, the rationale:**
5. Perl scripting for analysis of BGP data
6. Linux shell Scripting and tools for chopping huge files and reducing files for proper BGP data Analysis.
7. **References**
8. <http://www.routeviews.org/dynamics/>
9. <ftp://archive.routeviews.org/>
10. BGP archived data : <ftp://archive.routeviews.org/route-views.saopaulo/bgpdata/>
11. <http://www.mdzubaerhossain.com/home/zwiki/a-few-linux-commands>
12. <http://www.sedtutorial.com/>
13. <http://www.freeos.com/guides/lsst/ch04sec4.html>
14. http://routeviews.org/