Comcast Data Analysis

Suleyman Muhammad
Graduate Computer Engineer
Villanova Unversity Grad Class '11

Abstract

For my term project in CSC 8580 I have chosen to assist in the processing and analysis of consumer data extracted from a small subset of Comcast Cables consumer cable boxes. The selection of this project was partially due to my future employment with Comcasts NETO (National Engineering Technical Operations) division. The details of this project are as follows; I have been given a sample of recorded data from multiple consumers user boxes located within Pinole, CA. These measurements are directly pertaining to their VoD (Video on Demand) service which is a variable service, meaning usage varies with user demand (i.e. when a user deems necessary, they watch). This variability makes predicting consumer behavior fairly difficult. That is why Comcast has approached Villanova University with this task. While Comcast is efficient in implementation, Villanova University is much better suited in theoretical analysis. This fact is simply traditional in Academia and Comcast was intelligent enough to take advantage of this. My ultimate goal for this project was to perform a bit-rate aggregate in hopes to give way to some form of a trend model for particular times of day which will provide Comcast with a method for predicting the required Network Bandwidth for that particular timeslot. There were multiple intermediary objectives included with this project. The first objective, which occupied a majority of my time, was compression. In some ways this task was equally important as the final results because space efficiency allows for analysis of larger data sets. The compression portion of this project is explained in Section I. Once compressed my next sub-objective was extraction. Extraction is explained in more detail in Section II and mainly involved manipulation of the data set to a format better suited for my analysis. Finally I was able to begin my analysis of the data which is explained in Section III. This paper ends with a conclusion followed by the appendices which contain some of my source files used for implementation. I would appreciate being notified before any of my work is used further.

I Compression

As stated in the abstract compression took the majority of my time, not only in the aspect of implementation but also execution time. The compression script was written for the BASH shell on an Ubuntu 10.04 LTS Linux OS. To implement compression I took advantage of the columns of data which contained a great deal of redundancy. For example the providers column contained tens of thousands of rows containing the string PROVIDERS_XXXX where XXXX is the number corresponding to a particular provider. For Comcast, providers are essentially the television channels who provide different assets or television programs.

This same type of redundancy is found throughout the dataset. Parsing simply involved iterating through every variable of each line, searching for the string with the aforementioned redundancy then replacing it with an equally discernable string of shorter length. For example the string PROVIDERS_XXXX was replaced with P_XXXX. Proper utilization of the BASH internal field separator made it possible to separate every line into their relevant variables. To manage this process it was beneficial to treat the problem as a smaller subset of the total file. For example the script was first written to accommodate one record, then ten and so on until I could be confident the process would be effectively replicated over the entire data file. From a basic file size comparison I have determined that this compression has reduced the dataset size by around 16% which on a much larger dataset constitutes for a great deal. But as I will explain in the next subsection it is fairly unfeasible to do so because of the time efficiency of my BASH script. Alternatively the data set should be created with some thought into space efficiency excluding any unneeded redundancy.

The only downside as alluded to earlier is that the compression script 'compress.sh' ran for around 41 hours on a file of size 1.02 GB. This may seem inefficient, and very well may be, but in this case since the program should run only once efficiency is not necessarily an issue but improvements could definitely benefit the process. Some improvements that could affect my programs efficiency involve hard coding the particular columns which contain redundancy directly into the script. This should effectively make a majority of the string processing obsolete hopefully improving execution time. But before that statement can be made about the previously proposed improvement it would be important to determine which operation constitutes for my basic operation, comparisons or file manipulation. In other words the basic operation is the most demanding operation within the deepest nested loop, but that is a task for another time. I have included the compressed file in my deliverables.

II Extraction

The next step involved data extraction which was implemented using a very similar bash shell script titled extract.sh. The idea behind this script is the fact that for the final analysis I require the data to be in a format more suitable for Matlab, a high-level technical computing language and interactive environment for algorithm development which I have chosen to use to create an aggregate bit rate for analysis. This aggregate bit rate can then be utilized to classify particular trends for the data in question.

Extraction consisted of two parts. The initial task, as stated previously, was extracting the data variables into a format which could be easily processed in Matlab. A very important problem with the format of the data is that the start time and end time of each record is represented in a format readable for human eyes, but this is fairly useless for analysis. Luckily on my free time I had already written a C program which converts strings of characters representing a date and time into seconds calculated from January 1^st 1970. This time format is a standard in C and my calculations are very accurate. Once the start times were converted to seconds and useless character strings were removed from the data required I was then able to begin analysis.

The second part of extraction was directed towards the providers and their respective assets. To make the distinction between providers and their assets I simply placed a line within the extraction script which determined which provider the record belonged to and then appended this record to its respective file within the folder providers. This may seem very peculiar but I have been able to determine that the folder size is exactly the same size as my compressed data file. This step makes analysis of each provider much more strait forward. I have included this folder with my deliverables.

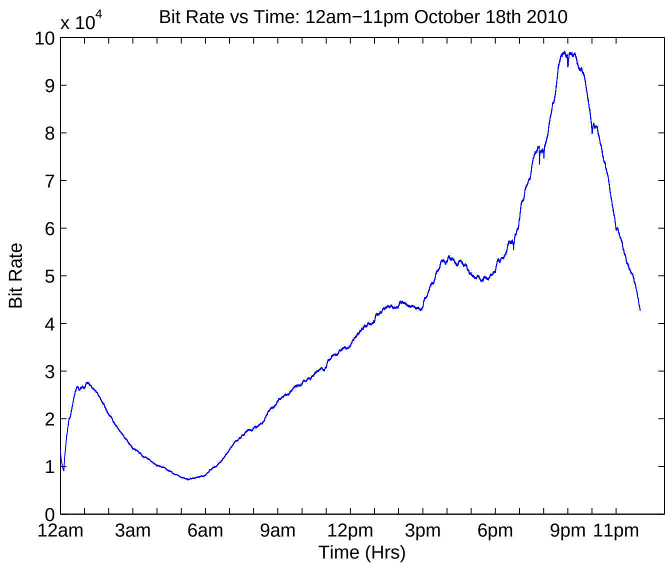

Fig. 1. Day 1

III Analysis

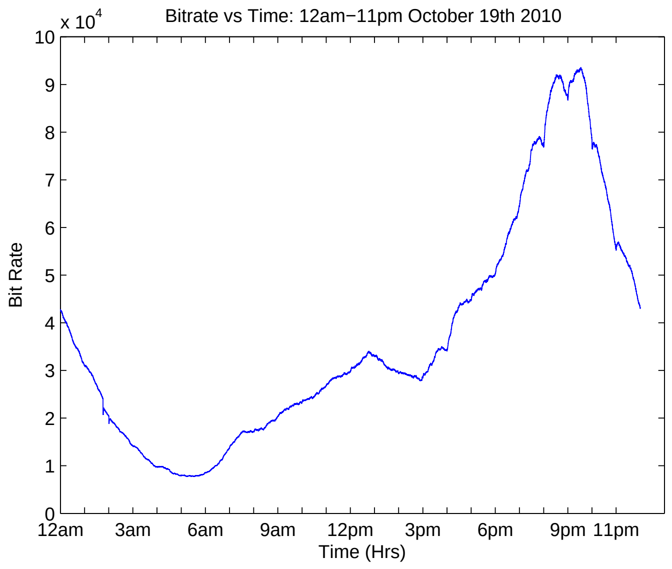

Fig. 2. Day 2

Although the actual analysis of the data is not entirely developed due to hardware/software limitations, my implementation is fully scalable. The aggregate contains only the data from Monday Oct. 18^th 2010 shown in Fig.1 and Tuesday Oct. 19^th 2010 shown in Fig.2. The limitations are due to various factors, one of which is the fact that the Matlab software I am running is located on a remote server and I have little control over memory usage and my aggregates precision is to the second. This puts a great burden on the program. For instance I was in fact able to load a $1.04 * 10^6$ by 8 data subset into the program and this subset was less than half of the total. With a personal license and administrator control over the system running the Matlab software I could possibly manage a more intuitive and all encompassing analysis of the week worth of data. Another option would be decreasing the amount of data points from second precision to minute, 15-minute, or hour precision.

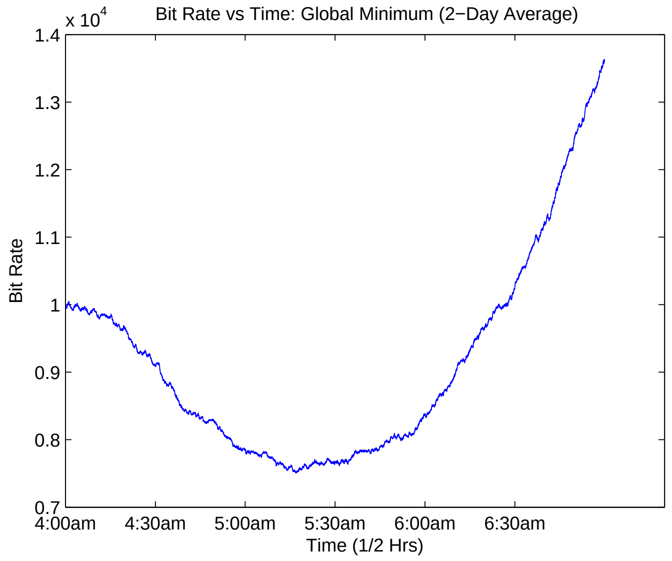

Fig. 3. Off Peak Time

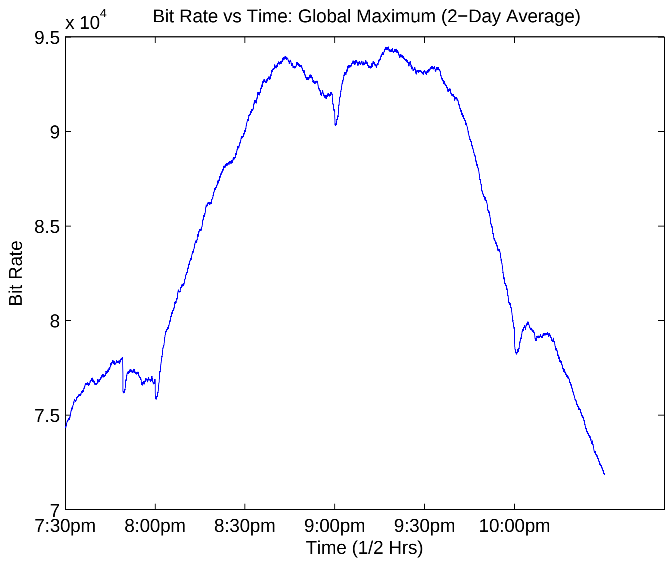

Fig. 4. Peak Time

With that being said I have still been able to provide some fairly interesting results from the bit aggregate of the two days. In both of these figures you can easily determine that there is a global minimum and maximum for required bandwidth at around 5:15am and 9:15pm shown in Fig. 3 and Fig. 4 respectively. The units have not been specified but I have estimated a minimum at $0.75 * 10^4$ total bit rate and a maximum at $9.45 * 10^4$ total bit rate. This results in a difference between minimum and maximum of $8.7 * 10^4$ units or 86,000 bit rate units. Keep in mind this is purely an estimate. With this information Comcast has the ability to predict the minimum required bandwidth for the peak time at around 9:15pm as well as how low they can scale this bandwidth during the period of low relative usage at around 5:15am. It must be noted that there is an anomaly in the data for Monday Oct 18th 2010 which is due to the fact that the recordings began at 12:00am and much of the aggregate that would have overlapped, say from 11:59 pm, is not included so the first hour or so is affected. This is also apparent in the 2-day average shown in Fig. 5.

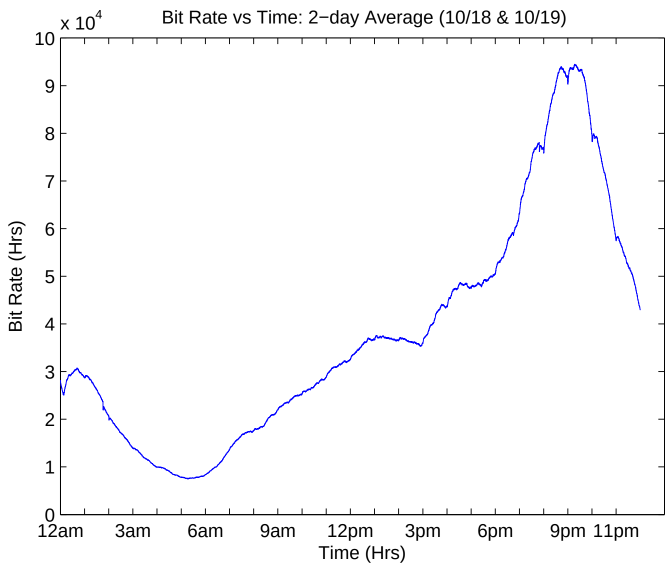

Fig. 5. Two Day Average

A very important addition to my analysis was pertaining to the providers although incomplete. In the Matlab file you will find a section in which I began the implementation of creating an aggregate bit rate for each provider, excluding which asset/assets provided what amount. The code that I have attempted to utilize is fundamentally sound but the time efficiency is of the same order magnitude as my compress.sh and extract.sh bash scripts. As explained previously nested loops provide a great deal of inefficiency. Because the providers are not already sorted within the data my code runs at around $\theta\left(n * n!\right)$ where n is the number of records. This can be easily seen with some trivial analysis of my methodology. A simple improvement would be to sort the data beforehand but because of time constraints I was unable to do so. I have included this within my mfile located in the appendix.

IV Conclusion

This was a very interesting project. I have definitely learned that often the most time consuming tasks are not necessarily the ones you originally expect. Although my analysis was greatly hindered by hardware and software limitations, my methodology is highly scalable. You can easily see from the aggregates where the low and peak times are for Comcasts VoD service. Given more capable hardware and possibly more time to manipulate the data it would definitely be possible to produce some very informative results. A simple example is a basic average of 30-days which could give maximum and minimum thresholds for a month of VoD data which can provide more reliable and robust results. If the opportunity presents itself I could extend my analysis to each provider and their respective assets similar to the methods detailed in Section II. Determining which providers provide the most and least content and which assets are provided both most and least frequently would be of insurmountable importance to those interested.

V Appendix - Mfile

Contents

CSC8580 Network Performance & Management -

```
%{
  Suleyman Muhammad
  Graduate Computer Engineer
  Villanova University Class '10
  Villanova Grad Class '11
  Final Project 2011 CSC8580: Network Management and Performance
%}
```

Initialization

```
clc;clear;close all

Submitted=datestr(now, 21)

cd('W:\Multimedia_GRAD\matlabCSC8580')

Submitted =

Apr.27,2011 14:11:25
```

Part 1a - load data

```
dat = csvread('cdat2.csv'); %Load in CVS file
[m n] = size(dat);
```

Part 1b - Initialize Storage For Aggregates

```
%Clear Unused Variables for use with aggregate
dat(:,5) = zeros(m,1);
```

Part 2 - Perform Aggregate Bitrate

```
for i = 1:m
   start   = dat(i,1); % Start time
   vsecs   = dat(i,2); % Vewing seconds
   bitrate = dat(i,3); % Bitrate
   for j = 1:(vsecs-1) % From the start time, for each veiwing sec
                       % increment the corresponding slot by the bitrate
       k=j-dat(1,1);
       %%AGGREGATE BITRATE
       dat((dat(i,1)+k),5) = dat(dat(i,1)+k,5) + dat(i,3);
   end
```

```
end

% Clear loop variables
clear i j k start vsecs bitrate;
```

Part 3 - Plot Data By Hour

```
half    = 30*60;     % Half hour in seconds
hour_s  = 60*60;     % One hour in seconds
day_s   = hour_s*24; % One day in seconds

day1=dat(1:day_s,5);          % Oct. 18th aggregate
day2=dat(day_s+1:2*day_s,5); % Oct. 19th aggregate
dayAVE = (day1+day2)/2;      % Average of 10/18/10 and 10/19/10

% Plot day 1 aggregate
figure (1)
plot(day1)
title('Bit Rate vs Time: 12am-11pm October 18th 2010')
 xlabel('Time (Hrs)')
 ylabel('Bit Rate')
 set(gca, 'XTick', 1:hour_s:day_s)
 set(gca, 'XTickLabel',{'12am','','','3am','','','6am','','','9am','',...
                        '','12pm','','','3pm','','', '6pm', '', '',...
                        '9pm','','11pm'})

% Plot day 2 aggregate
figure (2)
plot(day2)
title('Bitrate vs Time: 12am-11pm October 19th 2010')
 xlabel('Time (Hrs)')
 ylabel('Bit Rate')
 set(gca, 'XTick', 1:hour_s:day_s)
 set(gca, 'XTickLabel',{'12am','','','3am','','','6am','','','9am','',...
                        '','12pm','','','3pm','','', '6pm', '', '',...
                        '9pm','','11pm'})

% Plot 2-day average
figure (3)
plot(dayAVE)
title('Bit Rate vs Time: 2-day Average (10/18 & 10/19)')
 xlabel('Time (Hrs)')
 ylabel('Bit Rate (Hrs)')
 set(gca, 'XTick', 1:hour_s:day_s)
 set(gca, 'XTickLabel',{'12am','','','3am','','','6am','','','9am','',...
                        '','12pm','','','3pm','','', '6pm', '', '',...
                        '9pm','','11pm'})

% Plot Global Minimum
figure(4)
plot(dayAVE(4*hour_s:7*hour_s,1))
title('Bit Rate vs Time: Global Minimum (2-Day Average)')
```

```
 xlabel('Time (1/2 Hrs)')
 ylabel('Bit Rate')
 set(gca, 'XTick', 1:hour_s/2:3*hour_s)
 set(gca, 'XTickLabel',{'4:00am','4:30am','5:00am','5:30am','6:00am',...
                        '6:30am'})

% Plot Global Maximum
figure(5)
plot(dayAVE(19*hour_s+half:22*hour_s+half,1))
title('Bit Rate vs Time: Global Maximum (2-Day Average)')
 xlabel('Time (1/2 Hrs)')
 ylabel('Bit Rate')
 set(gca, 'XTick', 1:hour_s/2:3*hour_s)
 set(gca, 'XTickLabel',{'7:30pm','8:00pm','8:30pm','9:00pm','9:30pm',...
                        '10:00pm'})
```

Part 4 - Provider Analysis

```
%{
%Clear Unused Variables for use with aggregates
dat(:,6) = zeros(m,1);
dat(:,7) = zeros(m,1);

dat(1,7) = dat(1,8); %Initialize First Provider

k = 2;
no_match = 0;

for i = 1:m
   vsecs   = dat(i,2); % vewing seconds
   bitrate = dat(i,3);
   for j = 1:k % For each provider increment the corresponding slot
               % by the bitrate times the veiwing seconds
       if(dat(j,7) == dat(i,8))
           dat(j,6) = dat(j,6) + vsecs*bitrate;
       else if(j == k)
               no_match = 1;
           end
       end
   end

   if(no_match)
     k = k+1;
     dat(k,7) = dat(i,8);
     dat(k,6) = vsecs*bitrate;
     no_match = 0;
   end

end
%}
```