Modeling click-through rate of ads given the query and user information

Justin Thomas
Clemson University
School of Computing
Clemson, SC 29631
jt3@clemson.edu

Nikhil Prashant Bendre
Clemson University
School of Computing
Clemson, SC 29631
nbendre@clemson.edu

Dr.Amy Apon
Clemson University
School of Computing
Clemson, SC 29631
aapon@clemson.edu

ABSTRACT

Given the growth of e-commerce, search advertising has become one of the major revenue sources with the Internet industry. The most important feature of this economic model is to map the click-through rate (CTR) of ads, as this would help to rank the ads and the price clicks. The data set for this experiment was derived from session logs of the Tencent proprietary search engine soso.com. The data set was around 11GB. We found different trends based on the available user data and the search engine log data. The trend was to find out the click percentage based on depth of ad, the position of ad, different gender and age groups. We also performed linear regression on the whole data set. Analyzing this data was an intensive task and we worked on it using Hadoop Streaming and with parallel R – doMC package and data intensive package of R called bigmemory which is the part of bigmemory project. All the analysis was performed using Palmetto Cluster at Clemson University.

Author Keywords

Data Intensive Computing, Data Mining, Analytics, Cluster Computing

ACM Classification Keywords

Data Intensive Computing, Data Mining, Analytics, Cluster Computing

General Terms

Data Intensive Computing, Data Mining, Analytics, Cluster Computing

INTRODUCTION

The growing influence of the Internet in our everyday life has reached to such an extent that not only does the web content have to be good, but also the usability needs to be excellent. Estimates from 2006 indicate that the average U.S based user viewed 120 web pages per day. [1] The e commerce industry is flourishing very fast and now-a-days most of the sales take place via the Internet. In such a scenario, it has become important both for the consumer and the seller/marketer to benefit from better advertisements since most of the time people use search engines like Google, Yahoo, etc.

The market value of advertisements on the web totally depends on whether users click on displayed advertisements. There are many issues one can come across when trying to predict the probability of the user clicking an advertisement. Some of the issues that maybe related to advertisement are position of the advertisement, number of times it is displayed to the user in one session, advertisement relation with the query itself, etc. This project aims to study some of these factors as also along with coming up with an a sort of estimation or trend to show the user behavior associated with the advertisements.

DATA INFORMATION

The training file is of the size 9.9 GB and the user id profile file is 284 MB. The training data is a text file, where each line is an instance derived from a search session log message. A search session is an interaction between a user and the search engine. The terminology is as follows, the number of ads impressed in a session is known as the 'depth'. The order of an ad in the impression list is known as the 'position' of that ad. An Ad, when impressed, has the following properties such as title, a description, a keyword id, and an URL, known as a 'display URL'. The properties of the ad are ad id, unique ad id, and advertiser id, unique advertiser id. The click field denotes how many times the ad has been clicked. The query id field denotes the id of the query issued by the user in that particular search session. The last field in the training set is the user id field, which is also the key to the userid_profile data.

The userid_profile data set contains the user id field, the gender field which has possible values as 0 – unknown, 1 – male and 2 – female. It also has an age group field where we have 6 age group namely, ‘1’ for (0, 12], ‘2’ for (12, 18], '3' for (18, 24], ‘4’ for (24, 30], '5' for (30, 40], and '6' for greater than 40.

In the training file, wherever the user id information is not available, 0 value is used to denote that. Also, in the competition forum, there has been discussion about some

negative values for the Click field. However, upon testing the data set for all the unique values of the field Click it was found that there are no negative values. We have also replied on the forums and the competition administrator has also confirmed that there are no negative values.

BACKGROUND

Some of the technologies that we used were Hadoop Streaming using Python and Parallel R packages – bigmemory and doMC.

Large data sets, usually multi-gigabytes, often challenge and frustrate R users. The biggest challenge was to perform analysis on large data sets that don't fit in the working memory (RAM). Data frames and matrix object lack the flexibility and power of R's statistical programming, if they are used on such large data sets.

The big memory package provides an alternative by implementation of the big.matrix object that is similar to the matrix object in R, by using Boost C++ libraries. One of the main features is the ability of this data structure to be allocated to shared memory, thus allowing separate processes on the same or different computer to share access to a single copy of the data set. Even different parallel packages such as doMC, foreach, snow, and Rmpi can be used with these big matrices object thus allowing even naïve R users a chance to implement shared-memory parallel computing.

The doMC package is a "parallel backend" for the foreach package.[] The user however must register a parallel backend to provide the mechanism for foreach package to execute code in parallel or the code will be executed sequentially. The doMC package will perform the best when it runs on a machine with multiple processors or multiple cores or both. The doMC package doesn't have a stable release for the Windows platform. It only works with an operating systems that supports the fork system call.

Hadoop Streaming is a utility that comes with Hadoop distribution. The utility allows you to create and run map/reduce jobs with any executable or script as the mapper and/or the reducer[]. Basically, the mapper and reducer are executables that read input from stdin and emit output to stdout. So, the mapper will create a key/value pair, for each input line as mapper output. For example, the file name is key and the rest values are the value. The reducer will then read those key/value pairs and perform some utility functions like sort and emit the output to stdout. We have implemented one of the data preprocessing stages using the Python with Hadoop Streaming.

METHOD

We implemented four different strategies to analyze the big data,

1) Hadoop Streaming using Python

2) R - doMC package

3) R – bigmemory package

4) Programming with serial R scripts to verify the results

To start with we built custom virtual machines to be deployed on the Palmetto Cluster. Since, we were performing parallel multi core processing; we requested two single compute nodes with 8 cores for a wall time of 72 hours of different memory configurations, one with 16 GB and the other with 256 GB RAM. We had to download and install R-2.14.1on each of the nodes. We also had to mount the file system and download our data from the KDD Cup 2012 website which was a 2.8 GB zip file.

We wanted to analyze the click percentage based on the user profile details. The training file had Click information along with the user id field which served as the key in the userid_profile data set. To map these two files, we made use of the Hadoop Streaming using Python. The mapper read the input file, line by line and created a key/value pair. We also implemented an additional condition in the mapper. It checks whether the click value is 0 or not. We only want user ids from the input, which have a click value greater than 0, i.e only those users who have clicked on the ads at least once. Then the reducer reads those key/value pairs and then combines them and compares the user id field. If they match, a new line is emitted out to STDOUT and so on and we get a new data file. This method ensured that the new data file created had only three fields, user id, gender id, age group id.

Next we wanted to analyze, the click percentage based on Position and Depth fields. The unique possible values of Position and Depth fields were 1, 2 and 3. We read and stored the training data set in the big.matrix object using the big memory package in the R. The total numbers of rows were around 149 million. To implement the frequency count of the clicks, we developed a custom R function and invoked that function using the foreach and %dopar% functions. Here, we had to be careful since, because if we don’t register the number of cores then the %dopar% would be performed serially. So, registering doMC is the most important function call before any parallel operation. The function registerDoMC(n) would register the “parallel backend” and also specify the number of cores the program would run parallel.

One of the aims of the project was also to calculate the linear regression for the given data set. The dependent variable was “Click” field and the independent variables were “Position” and “Depth” fields. Since, the data set was 9.9 gb, we had to implement it using the biglm package available in R. The biglm package is a part of the biganalytics package in R.

ANALAYSIS

Given the search engine log data, we found the following trends.

1. Click % based on Position
2. Click % based on Depth
3. Click % based on User Age Group
4. Click % based on User Gender
5. Prediction of Click through Rate using linear regression.

1. Click % based on Position

From all the log data, we filtered out the click types based on position.

Position	Clicks	Non-Clicks
1	4928263	83873377
2	1524750	46913818
3	205023	12193874

Table1: Click Data based on Position

The technique we used to find out the estimation was how many clicks took place given the total number of clicks and non-clicks. The following graph indicates the trend.

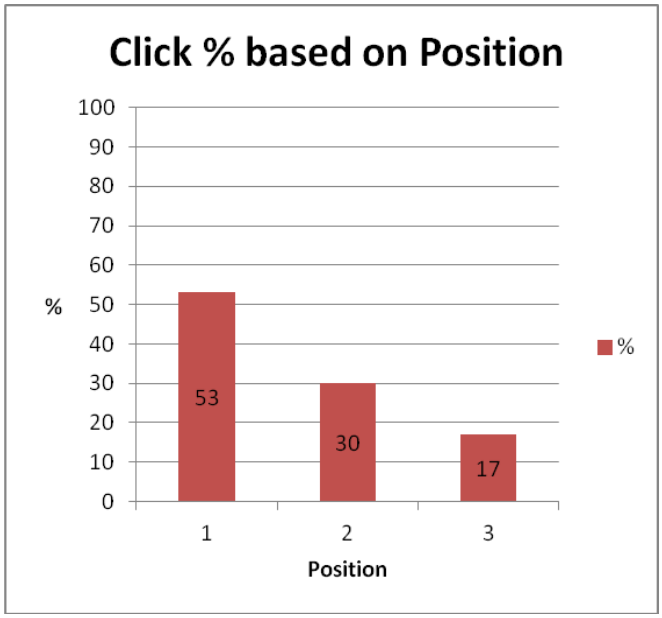

Figure 1: Estimation of Click % per Position

We can see that the results indicate that Position 1 gives the maximum percent of the clicks based on the position followed by Position 2 and 3. Nearly, there is 50% difference between Positions. This is an important result since the search engine company can decide the display price of the ad now based on the Position. Higher the position the more likely it is going to be clicked.

2. Click % based on Depth

From all the log data, we filtered out the click types based on depth.

Depth	Clicks	Non-Clicks
1	1974457	39542107
2	3594848	68009638
3	1088731	35429324

Table2: Click Data based on Depth

We used a similar technique as the Position one to find out the estimation of how many clicks took place given the total number of clicks and non-clicks. The following figure 2 indicates the trend. The depth indicates how many times the user was shown different advertisements.

The surprising fact as we can see in the Figure 2 is that the chance of the user clicking the advertisement is not dependent on the depth. This result could aid the advertiser since the advertiser would benefit only from the position not on how many types it is displayed to the user. So, the ad value should not be decided on how many times it is shown since the Click percentage is nearly the same for Depth values 1 and 2.

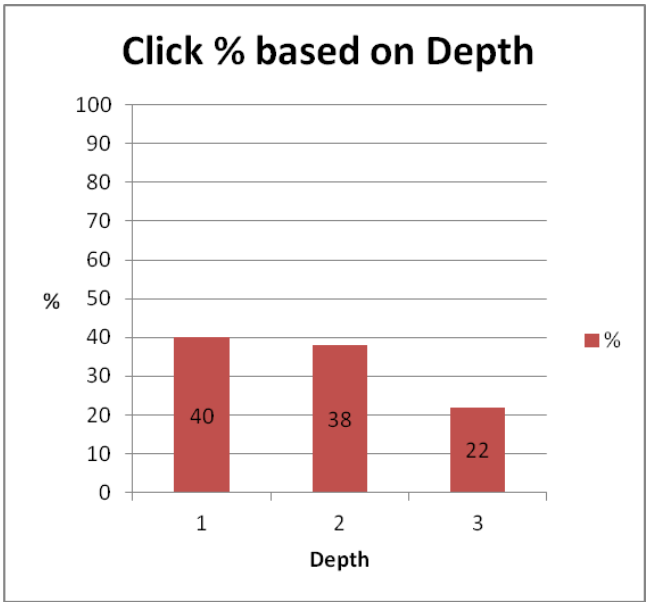

Figure 2: Estimation of Click % per Depth

3. Click % based on Age Group

From all the log data, we got after the merge operation; we filtered out the click types based on user age group.

Age Group	Clicks	Non-Clicks
1-12	439009	8378646
13-18	892921	17061167
19-24	1685001	33905684
25-30	1149253	23652811
30-40	867053	15725037
40+	417912	6741727

Table3: Click data based on Age Groups

With this information, we can find interesting click % based on this demographic information.

From the graph, we can see that people from age group 40+ click on the ads the most and surprisingly the people from the age group 25-30 click the least on the ads.

There is a possibility where people have provided false data about their age, but there was no way we could have verified this factor, since the given data set just consists of the age groups and not the individual age of each user.

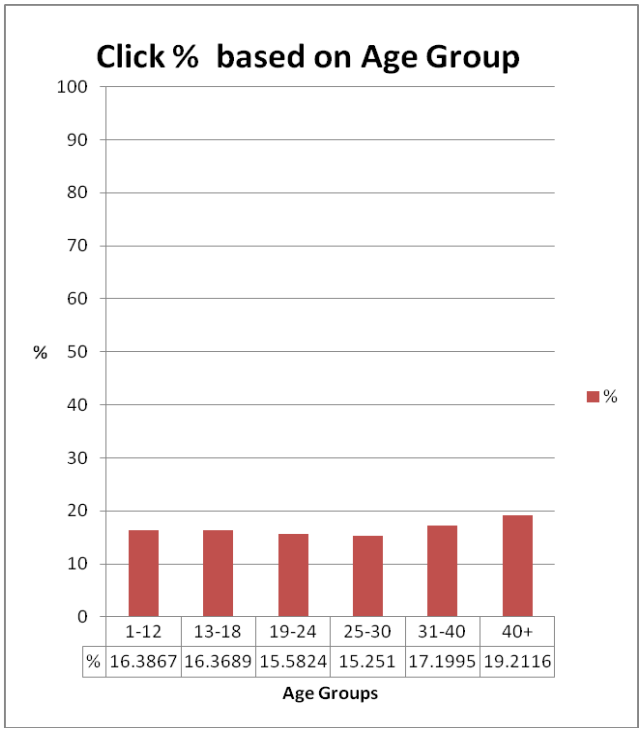

Figure 3: Click % based on Age Groups

4. Click % based on Gender

From all the log data, we got after the merge operation; we filtered out the click types based on user gender.

Gender	Clicks	Non-Clicks
Male	2951284	58193003
Female	2424299	45842498
Unknown	75566	1429571

Table4: Click data based on Gender

From the graph, we can see that the Click % based on user gender is not conclusive. The percentage of female who have clicked on the ads is bit more than percentage of males but the difference is just about 1%.

Also, there is a factor of unknown in the data set which is nearly equal to the male and female percentage. If this data would have been just male and female data then we could have got a much better result.

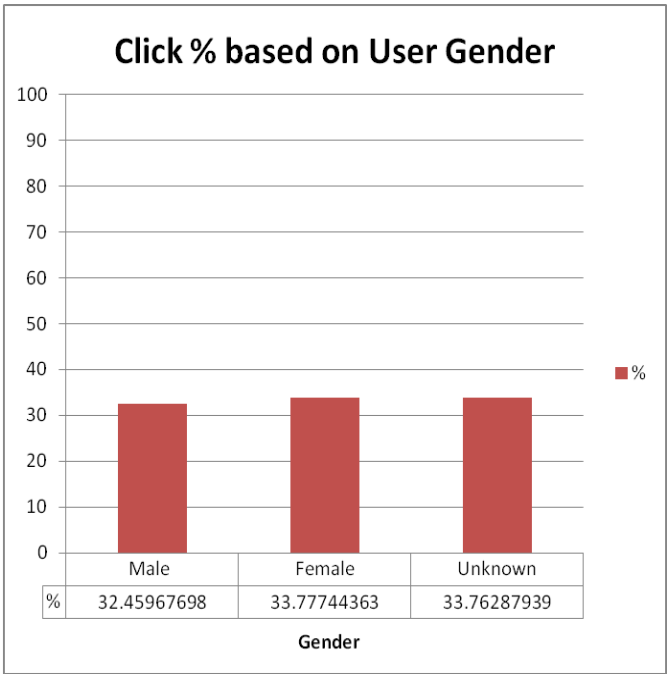

Figure 4: Click % based on Gender

5. Prediction of Click through Rate using linear regression

We performed the linear regression on the entire data set, the sample size being 149 million records. The independent variables were Position and Depth and the dependent variable was the Click field.

On inspecting the p value, we can see that p=0.7191 for Depth and p=0.0000 for Position. For the result to be

significant, the p value should be less than 0.05. (p<=0.05). In that case, only Position field really affects the Click value. The coefficient value of position is -0.0291 which says indicates that increase in Position value will decrease the No of Clicks for that position. So, lower the position value, i.e position 1 will get more clicks than position 2 and 3. This theory can also be verified from the results that we have got in the Click percentage based on Position.

This result could be important from the advertiser's perspective since he has this information and plans accordingly to buy ads in that position. Similarly, the search engine could also charge a premium price for ads at higher position.

	Coef	95%	CI	P
(Intercept)	0.0988	0.0959	0.1016	0.0000
Depth	-0.0003	-0.0019	0.0013	0.7191
Position	-0.0291	-0.0308	-0.0273	0.0000

Table 5: Linear Regression Result

Performance Benchmarks

We measured performance on the Palmetto Cluster for different configuration nodes. One node was the bigmem node which was an Intel Xeon 7542, clock speed of 2.66 GHz with 256 GB memory. The other node was an Intel Xeon L5420, clock speed of 2.5Ghz with a memory of 16 GB. Both the compute nodes had 8 cores as the core configurations.

Node	Processor	Cores	Memory
Node 1188	Intel Xeon L5420@2.5GHz	8	16GB
Nodelm03	Intel Xeon 7542@2.66GHz	8	256GB

Table 6: Compute Node Hardware Specifications

Amdahl's Law is often used in parallel computing to predict the theoretical maximum speedup using multiple processors.

This performance analysis was done considering only the bigmem machines. Serial R operations could not be performed on the workq (16GB) machines since the read.table( ) call went out of working memory. So, the data could not be loaded and hence there was no way we could perform the comparison between serial and parallel operations involving the workq 16 GB machines.

We calculated maximum speedup by taking the ratio of the time taken for serial and parallel operation. We performed analysis on 8 cores, so ideally the parallel operations should be 8 times faster than the serial operations. But practically, we only got a speedup of 6. Possible reasons could be the file system or the memory cache of the compute node.

By Amdahl's law, the speedup can be calculated as S(N) = S+P/S+(1/N)P, where S= serial part, P = parallel part and N=Number of processors. So, substituting all the values, 6 = 1/S+8(1-S), S = 1.11, so P = 100-S = 98.89% parallelizable.

The maximum theoretical speedup is (1/S), that is, 90.09.

Operation	Serial	Parallel
Read	59.88 mins	10.18 mins
Linear Regression	45.8 mins	7.3 mins

Table 7: Performance of Read & Linear Regression Operations

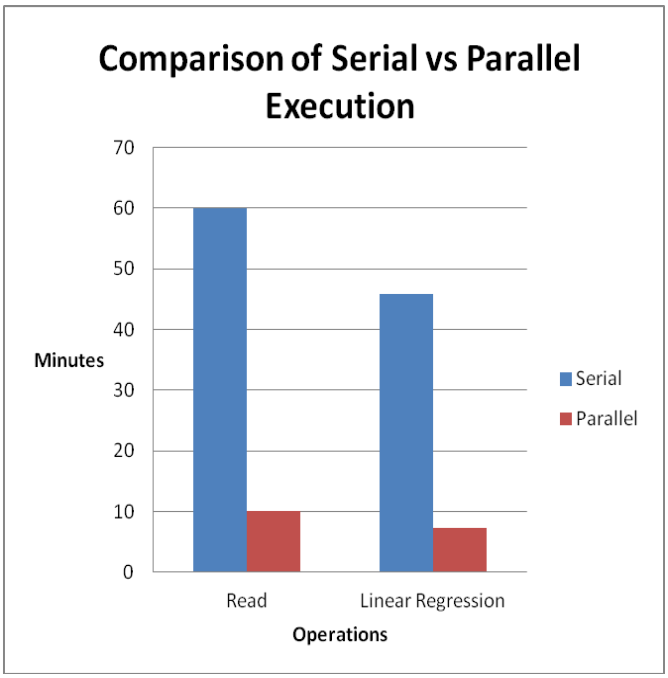

Figure 5: Performance Comparison of Serial vs Parallel Execution I

For Aggregation operations, the speedup differed from 2 to 6 times the serial operations. The possible reasons could again be the file system or the memory cache or even the way we implemented the parallel operation. We have performed all the parallel operations using the big matrix and the doMC packages.

Operation	Serial	Parallel
Count 0 Clicks per Position	29.3 secs	17.3 secs
Count 0 Clicks per Depth	28.75secs	17.3 secs
Count of all instances of Clicks/Position	25.67 secs	4.6 secs
Count of all instances of Clicks/Depth	25.23 secs	4.9 secs

Table 8: Performance of Aggregation Operations

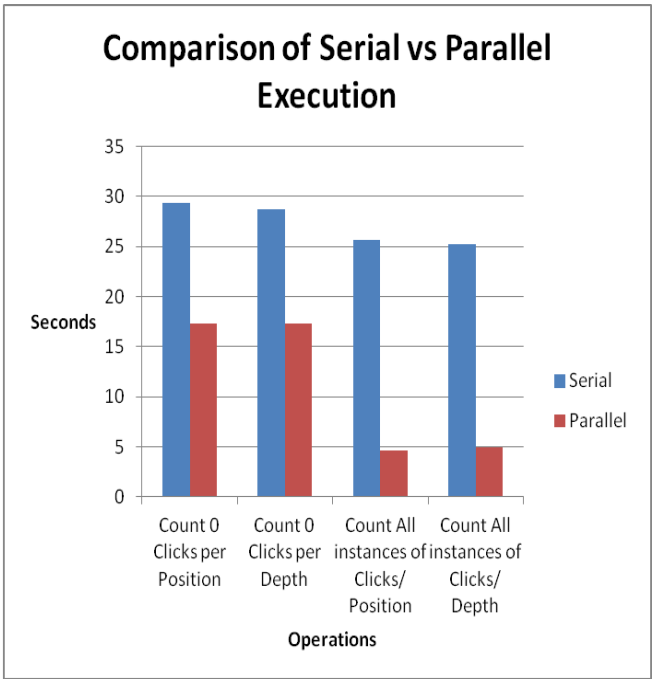

Figure 6: Performance Comparison of Serial vs Parallel Execution II

CHALLENGES

We encountered challenges at every other point in this project. We could not even open our file since the file was around 10 GB and even with a small userid_profile file which was around 285 MB we ran out working memory as soon as we tried to do any kind of analysis on it. When we tried to read data using the read.table() function in R, we got an error message which was "Error: cannot allocate vector of size 2.0 gb" .

We overcame the working memory challenge by using the bigmemory and doMC package. But since the bigmemory project is quite new, there was hardly any documentation other than manual which was released this month April 2012!

There was no direct merge operation in bigmemory package since it is built on Boost C++ library. To overcome this merge problem, we resorted to using Hadoop Streaming using Python, and then wrote the map-reduce program to get the necessary data in a separate file.

To perform linear regression, we wanted to use the whole data set. Other way was to make smaller files and then calculate linear regression values but then it wouldn't have served any purpose. There was biglm function as a part of the big analytics package available in the bigmemory project. This was also one of the reasons why we choose bigmemory package.

To implement the doMC package on Palmetto, we had to install custom virtual machines on each node and then setup R libraries and packages and then perform all the analysis. There is no direct way to implement doMC package on Palmetto since it is multicore parallel programming.

CONCLUSION

This project taught us different strategies to deal with big data. We demonstrated different techniques for performing analysis on huge data sets and implemented those using Hadoop Streaming and Parallel R packages. We modeled the user clicks based on the search engine log data and these results demonstrated that Clicks were more dependent on Position rather than the Depth of a given ad.

The performance results based on serial and parallel execution of a bigmem node gave a speedup of 6 instead of the theoretical speedup of 8. This would help us to investigate the problems with the memory or the hardware infrastructure of the Palmetto Cluster.

FUTURE WORK

This project can be further extended by performing some data mining stratergies like classification. The data set could be extended to include the additional data files and more aggregation operations could be performed on them.

ACKNOWLEDGMENTS

We would like to place on record our sincere gratitude to Pengfei Xuan and Dr.Lihn Ngo for helping us out with R and the PBS scripts. We would also like to thank Dr.Amy Apon for guiding us throughout the semester.

REFERENCES

1. Georg Buscher, Edward Cutrell, and Meredith Ringel Morris. 2009. What do you see when you're surfing?: using eye tracking to predict salient regions of web pages. In *Proceedings of the 27th international conference on Human factors in computing systems* (CHI '09). ACM, New York, NY, USA, 21-30. DOI=10.1145/1518701.1518705