# ***Contextual Advertising***
# ***Apache Hive and Amazon EMR***

# ***Storing Logs on Amazon S3***

```
s3://elasticmapreduce/samples/hive-ads/tables/impressions/
   dt=2009-04-13-08-05/ec2-12-64-12-12.amazon.com-2009-04-13-08-05.log
```

# ***Launching a Development Job Flow***

```
ssh -i ~/my-keypair-private-key.pem hadoop@ec2-67-202-12-120.compute-1.amazonaws.com
```

# ***Running a Hive Session on the Master Node***

```
hive \
   -d SAMPLE=s3://elasticmapreduce/samples/hive-ads \
   -d DAY=2009-04-13 -d HOUR=08 \
   -d NEXT_DAY=2009-04-13 -d NEXT_HOUR=09 \
   -d OUTPUT=s3://test-data-services

ADD JAR ${SAMPLE}/libs/jsonserde.jar ;
```

# ***Declaring Tables in Amazon S3***

```
CREATE EXTERNAL TABLE impressions (
   requestBeginTime string, adId string, impressionId string, referrer string,
   userAgent string, userCookie string, ip string
 )
 PARTITIONED BY (dt string)
 ROW FORMAT
   serde 'org.apache.hive.hcatalog.data.JsonSerDe'
   with serdeproperties ( 'paths'='requestBeginTime, adId, impressionId, referrer, userAgent, userCookie, ip' )
 LOCATION '${SAMPLE}/tables/impressions' ;


ALTER TABLE impressions ADD PARTITION (dt='2009-04-13-08-05') ;
MSCK REPAIR TABLE impressions


CREATE EXTERNAL TABLE clicks (
   impressionId string
 )
 PARTITIONED BY (dt string)
 ROW FORMAT
   SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
   WITH SERDEPROPERTIES ( 'paths'='impressionId' )
 LOCATION '${SAMPLE}/tables/clicks' ;

MSCK REPAIR TABLE clicks
```

# Combining the Clicks and Impressions Tables

```
CREATE EXTERNAL TABLE joined_impressions (
  requestBeginTime string, adId string, impressionId string, referrer string,
    userAgent string, userCookie string, ip string, clicked Boolean
  )
  PARTITIONED BY (day string, hour string)
  STORED AS SEQUENCEFILE
  LOCATION '${OUTPUT}/joined_impressions'
 ;

CREATE TABLE tmp_impressions (
  requestBeginTime string, adId string, impressionId string, referrer string,
  userAgent string, userCookie string, ip string
 )
 STORED AS SEQUENCEFILE;


INSERT OVERWRITE TABLE tmp_impressions
  SELECT
    from_unixtime(cast((cast(i.requestBeginTime as bigint) / 1000) as int)) requestBeginTime,
    i.adId, i.impressionId, i.referrer, i.userAgent, i.userCookie, i.ip
  FROM
    impressions i
  WHERE
    i.dt >= '${DAY}-${HOUR}-00' and i.dt < '${NEXT_DAY}-${NEXT_HOUR}-00'
 ;

CREATE TABLE tmp_clicks (
  impressionId string
 ) STORED AS SEQUENCEFILE;

INSERT OVERWRITE TABLE tmp_clicks
  SELECT
    impressionId
  FROM
    clicks c
  WHERE
    c.dt >= '${DAY}-${HOUR}-00' AND c.dt < '${NEXT_DAY}-${NEXT_HOUR}-20'
 ;


INSERT OVERWRITE TABLE joined_impressions PARTITION (day='${DAY}', hour='${HOUR}')
 SELECT
  i.requestBeginTime, i.adId, i.impressionId, i.referrer, i.userAgent, i.userCookie,
  i.ip, (c.impressionId is not null) clicked
 FROM
  tmp_impressions i LEFT OUTER JOIN tmp_clicks c ON i.impressionId = c.impressionId
 ;
```

# Running in Script Mode

s3://elasticmapreduce/samples/hive-ads/libs/join-clicks-to-impressions.q

```
 $ SAMPLE=s3://elasticmapreduce/samples/hive-ads
 $ OUTPUT=s3://mybucket/samples/output
 $ ./elastic-mapreduce --create --name "Join Clicks" \
    --hive-script --arg $SAMPLE/libs/join-clicks-to-impressions.q \
    --args -d,SAMPLE=$SAMPLE \
    --args -d,DAY=2009-04-13,-d,HOUR=08 \
    --args -d,NEXT_DAY=2009-04-13,-d,NEXT_HOUR=09 \
    --args -d,INPUT=$SAMPLE/tables \
    --args -d,OUTPUT=$OUTPUT \
    --args -d,LIB=$SAMPLE/libs
```

# Contextual Advertising Model

# to estimate the probability of a click given the context.
 P[click|context]

# One heuristic for doing this is the following formula.
 product_{f in context} Pr[click|f=true]

# This heuristic multiplies the probability of a click for each feature that is true in the advertising context. If we take the negative log of this formula, we get the following formula.
 - sum_{f in context} log (count[click,f=true] / count[f=true])

# Declaring External Tables in the Interactive Job Flow

```
hadoop@domU-12-31-39-07-D2-14:~$ hive \
   -d SAMPLE=s3://elasticmapreduce/samples/hive-ads
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS MODEL_joined_impressions (
  request_begin_time string, ad_id string, impression_id string,
  page string, user_agent string, user_cookie string, ip_address string,
  clicked boolean
 )
 PARTITIONED BY (day STRING, hour STRING)
 STORED AS SEQUENCEFILE
 LOCATION '${SAMPLE}/tables/joined_impressions';


MSCK REPAIR TABLE MODEL_joined_imressions;

SHOW PARTITIONS MODEL_joined_impressions;
```

# Producing the Feature Matrix

# transformation on our impression data to produce Boolean features

# User Agent
#———————

# convert the user agent string into a sequence of keywords is to use a python script. As we'll see shortly, we can call this script directly from within a Hive statement.

```
  #!/usr/bin/python

  import sys
  import re

  for line in sys.stdin:
    user_agent, ad, clicked = line.strip().split('\t')
    components = re.split('[;/,\(\) ]', user_agent)
    for component in components:
      if len(component) != 0:
         print '\t'.join([component, ad, clicked])
```

```
# call this script from within a Hive, we issue a MAP statement.
  MAP
   MODEL_joined_impressions.user_agent, MODEL_joined_impressions.ad_id,
   MODEL_joined_impressions.clicked
  USING
   '${SAMPLE}/libs/split_user_agent.py' AS
   feature, ad_id, clicked
  FROM
    MODEL_joined_impressions
  LIMIT 10;
```

```
<code_snippet>
Query ID = hadoop_20210316030721_9f7629e0-cf49-4676-94ea-027fc85b66a1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1615776600752_0017)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED     12         12        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 21.76 s
----------------------------------------------------------------------------------------------
OK
Mozilla  u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
5.0         u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
Windows             u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
U           u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
Windows             u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
NT          u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
5.1         u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
en-US    u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
rv:1.9.0.10         u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
Gecko    u375QMdkb3BaP0Kv4QaQTFoxg587nV     true
Time taken: 23.352 seconds, Fetched: 10 row(s)

</code_snippet>
```

# The columns user_agent, ad_id, and clicks from the joined_impressions table are input to the script and the result is a table with the columns feature, ad_id, and clicked.
# The output of the statement is displayed on the console so we limit the number of lines output to ten. We can see from the output that the keywords contain spaces and are not lower cased. To normalize the output we

apply the user defined functions trim and lower and we prefix each keyword by 'ua:' so these features can be mixed with other features.

```
  SELECT concat('ua:', trim(lower(temp.feature))) as feature, temp.ad_id, temp.clicked
  FROM (
    MAP MODEL_joined_impressions.user_agent, MODEL_joined_impressions.ad_id,
MODEL_joined_impressions.clicked
    USING '${SAMPLE}/libs/split_user_agent.py' as feature, ad_id, clicked
    FROM MODEL_joined_impressions
  ) temp
  LIMIT 10;
```

# IP Address

```
SELECT
    concat('ip:', regexp_extract(ip_address, '^([0-9]{1,3}\.[0-9]{1,3}).*', 1)) AS
      feature, ad_id, clicked
  FROM
    MODEL_joined_impressions
  LIMIT 10;

OK
ip:45.206           AAv8arUW6Uw8HsXfssxearjVRbIOU9      false
ip:43.201           S15U6hxbUmrFNdJvQLi9KtQDRlwkvo      false
ip:43.189           mgH3jRl4TKjU0pwNCHTu062sMUkMi7      false
ip:32.32 le4T9koKQPnHjWKixqrR6ofwDSotrW       false
ip:21.45 LVTkiC1ThKHde4JjmkfrGqnigMVUeM       false
ip:34.192           d1M9EmKxHfLsOb5xG94X6vWVfXtGMm              false
ip:61.87 GOr35CvAg0dGE4QcQE8ChJMPRMpc0o  false
ip:20.198           VgsaJ62M6D7r0apuCSpmaLQ1d66qQg      true
ip:66.205           rRbJTRwfru8IpTMsDteTOIhU984Vji        false
ip:40.175           d1M9EmKxHfLsOb5xG94X6vWVfXtGMm              false
Time taken: 1.155 seconds, Fetched: 10 row(s)
```

# URL
#———

# To extract a feature from the URL of the page on which the advertisement displays, we make the URLs all lowercase and add "page:" to the beginning.

```
  SELECT concat('page:', lower(page)) as feature, ad_id, clicked
  FROM MODEL_joined_impressions
  LIMIT 10;

OK
page:yimg.com    AAv8arUW6Uw8HsXfssxearjVRbIOU9      false
page:schuelervz.net          S15U6hxbUmrFNdJvQLi9KtQDRlwkvo      false
page:21cn.com    mgH3jRl4TKjU0pwNCHTu062sMUkMi7   false
page:ibibo.com   le4T9koKQPnHjWKixqrR6ofwDSotrW      false
page:barnesandnoble.com  LVTkiC1ThKHde4JjmkfrGqnigMVUeM    false
page:paipai.com  d1M9EmKxHfLsOb5xG94X6vWVfXtGMm             false
page:startimes2.com          GOr35CvAg0dGE4QcQE8ChJMPRMpc0o  false
page:indiatimes.com          VgsaJ62M6D7r0apuCSpmaLQ1d66qQg      true
page:gougou.comrRbJTRwfru8IpTMsDteTOIhU984Vji          false
page:gougou.comd1M9EmKxHfLsOb5xG94X6vWVfXtGMm             false
```

```
Time taken: 0.127 seconds, Fetched: 10 row(s)
```

# Combining the Features
# Now that we've written queries to normalize each of the feature types let's combine them into one table. We can do this using Hive's UNION operator. Keep in mind that all sub queries in the union must have the same number of columns that have the same, exact names.

```
  SELECT *
  FROM (
    SELECT concat('ua:', trim(lower(ua.feature))) as feature, ua.ad_id, ua.clicked
    FROM (
      MAP MODEL_joined_impressions.user_agent, MODEL_joined_impressions.ad_id,
MODEL_joined_impressions.clicked
      USING '${SAMPLE}/libs/split_user_agent.py' as (feature STRING, ad_id STRING, clicked BOOLEAN)
      FROM MODEL_joined_impressions
    ) ua

   UNION ALL

   SELECT concat('ip:', regexp_extract(ip_address, '^([0-9]{1,3}\.[0-9]{1,3}).*', 1)) as feature, ad_id, clicked
   FROM MODEL_joined_impressions

   UNION ALL

   SELECT concat('page:', lower(page)) as feature, ad_id, clicked
     FROM MODEL_joined_impressions
   ) temp
   limit 50;
```

# Note that we had to modify the user agent query slightly. Passing data through a mapper strips the columns of their types and returns them as strings. To merge with the other tables, we need to define clicked as a Boolean.

# Index Table

```
CREATE TABLE feature_index (
    feature STRING,
    ad_id STRING,
    clicked_percent DOUBLE )
  STORED AS SEQUENCEFILE;


INSERT OVERWRITE TABLE feature_index
    SELECT
      temp.feature,
      temp.ad_id,
      sum(if(temp.clicked, 1, 0)) / cast(count(1) as DOUBLE) as clicked_percent
    FROM (
      SELECT concat('ua:', trim(lower(ua.feature))) as feature, ua.ad_id, ua.clicked
      FROM (
        MAP MODEL_joined_impressions.user_agent, MODEL_joined_impressions.ad_id,
MODEL_joined_impressions.clicked
        USING '${SAMPLE}/libs/split_user_agent.py' as (feature STRING, ad_id STRING, clicked BOOLEAN)
      FROM MODEL_joined_impressions
    ) ua
    UNION ALL

    SELECT concat('ip:', regexp_extract(ip_address, '^([0-9]{1,3}\.[0-9]{1,3}).*', 1)) as feature, ad_id, clicked
    FROM MODEL_joined_impressions

    UNION ALL

    SELECT concat('page:', lower(page)) as feature, ad_id, clicked
    FROM MODEL_joined_impressions
  ) temp
  GROUP BY temp.feature, temp.ad_id;

cast(count(clicked = 'true') as DOUBLE)

sum(if(clicked = 'true', 1, 0))
```

# Applying the Heuristic

```
SELECT
    ad_id, -sum(log(if(0.0001 > clicked_percent, 0.0001, clicked_percent))) AS value
  FROM
    feature_index
  WHERE
    feature = 'ua:safari' OR feature = 'ua:chrome'
  GROUP BY
    ad_id
  ORDER BY
    value ASC
  LIMIT 100
  ;


---------------------------------------------------------------------------------------------
OK
9DGhVCJPfCkoXA1jJ3UtHGWfU0whlL    0.9808292530117262
k60lJjN5sr3JxT9tOgCRA29jgow7CX    1.0986122886681098
IcJq5iHUCXPgMVPfIGfxFC3XhbiRJj    1.2321436812926323
heG6lG11EQwGgIQStm2dv0M6qe3saP    1.252762968495368
RVtOSibJxch3OJoEELE3RMlUC2jVKc    1.2580400255962119
mAvGKX705nFv35rennhwEqL6BGJXQM  1.349926716949016
eEPpmatr12b3IwfGwbGfo2fPDbUpIm    1.4226620052907655
7XkCp8ndNPQaH88GTwsReE8qNqlfff    1.4350845252893225
XshiIrb6d7S8IEd25bPIq1trJNauuK1.4350845252893225
3uk7HODLcmubKbQu7uDHoupTrHv21l    1.4469189829363254
UXEIPrcKVoePINjvhg7BeRj2utWNjc    1.4816045409242156
Qax0xaEx1u44DeJP0OwAPRKQ41QUs9    1.4816045409242156
73oEqx3Mkk88apIpLwp02IgroItcHt    1.5260563034950494
ubKkdWnDqUNjaaH5Mh2tLfhGeO4gkg    1.5686159179138452
4pGQfqJWhTcgNFCmGIqaJjEcipI49n    1.6582280766035324
fe02VgK4oFJp4hcgHSmxthJa9xM3HP    1.6582280766035324
IdW1dwPIBh7WdKCxE3Wljlik06J6vR    1.7047480922384253
7ORhspStIoX59Mh3Io2FLw8NbBLcEA    1.7272209480904839
wuanC2Q3uOFjF2vIe1UsDSAO4IaCgv    1.7346010553881064
35o4b04xEPh5l4ts1p4jlAsQkGSR9i    1.7491998548092589
6Puw5UL9LwWdfb01192Jq2mCwfITUB    1.749199854809259
OSk5qN7upGVHjsC8VmTae0W6jkTLiW    1.776011112259916
```

3Efxu1QEJPi0ES1oPKpUT31wKxgs1C	1.791759469228055
MliHGihLh6V0Ci4oMifpORQXu5pU6m	1.791759469228055
lXkhO5fdhb7c3NIkC9kHP4Sv4uA4aG	1.791759469228055
aJXiRii3hsQ8FrHqWogKRinnixTvhX	1.791759469228055
ihXF7HBq62LK7suwDFguW4qG0TvoGU	1.791759469228055
3LUsLrdCj4sRSDEUNsIg7MwXTkaLMx	1.791759469228055
v6GTMOrSSTseBP50bT8tNHUxLADagj	1.791759469228055
9eO9paWEuuT04S06HTqP8eGpAw3197	1.8137383759468304
FiNO6TnQpg4n03fvv61whuDsHjJqwq	1.8325814637483102
13QnG1wqVSj64LJo5RpJuaX5grRVhN	1.8325814637483102
lIXlSS5vjcaBpawl1vQV3dfGwI5UxG	1.8458266904983307
8Lcc413EqE2BNbC2kx1aLRJfKlQeKV	1.8458266904983307
I5FhtPH1XepqgkPBBGWhVIne6BpgjW	1.845826690498331
jVKbCxOWq4WEQFVC3hoMffWcho3KwF	1.845826690498331
HXXvt8irJJfQJlJ2HUU9lJfohtnQBG	1.845826690498331
r5JHuEtIOHWonBfcrvFveHoGHtpS9R	1.845826690498331
lV3T61466Su7d3e722AqnWpvuKNmgh	1.8523840910444898
lKLmMmRo4bUvArVJwX1UPvCX9Pi6jS	1.856297990365626
n25ash1v0BG5sX0kFmWdvdlU3UQ8St	1.8718021769015913
mVVHohs0I5ejIraS16mPs652mGQ11O	1.8971199848858813
oCeuObOGV9RV9K9HnVT8ChgCDfaAAt	1.9095425048844383
1jWd7MewucGb8joT1altmLsehCJkDH	1.916922612182061
dlDkwPFeRxE6dKSUMPOnDMAaUajKQT	1.929909807708872
l8WaV2TvvtGdURckeMAGEcRacdIu48	1.9315214116032138
sSP4gEKCUhEruOxS8WcOoIKRq6FMBm	1.9459101490553135
2s4geni54iw4QauLC1U5FRf8xH2ePJ	1.9459101490553135
JRN4RBXsC4QPdEDE881uPhAOJk4VpR	1.9459101490553135
9gl50aO3ufeBagOfnm3tW8Ob8bO0kX	1.9459101490553135
49BQTWdeNW7f9XqSxkTNRBf2FS7Wph	1.9459101490553135
KghFbdpcOEed6aKaiEelqqPSpuqXNl	1.9503644994046936
dVS09w0SFNPFq6vXI7BBcxnOTeii5W	1.992430164690206
95JUFwJtrPqffPNVUBCOKN9dNTCh3x	1.9924301646902063
5Ngeh0duRrigoMxUhr8ItoxBG2FDsk	2.0149030205422647
nUCBo2CTdvmkSTQWHBQ80NUpXej1Dv	2.0149030205422647
k828HBVrEXUlVI6nPR3DHKsxR1dMpk	2.0149030205422647
6tgVI9eJDBPPtwLh3b7CubpC1MnleV	2.0281482472922856
04x4hNC8jQEoP490vrM3Lvr9h61CMX	2.0281482472922856
ssRQ75fpPbsIu9IdVPCIE1qNRRon8N	2.0281482472922856
HsodC44GXjGWDfWSAw1Tb7EKruNU2j	2.03688192726104
vo3I6c560k9MliqjsbRXvaFfNiGBkn	2.03688192726104
IuB9hK22dL4cGgcrvEFxSmSECXjCLS	2.043073897508961
opAmlcxjUjclxJvXaedH0VMCKbhAbJ	2.063693184711697
u63dGD1UTBA1RSkCLekHNf9vgrQ8op	2.0794415416798357
1OtO10LorFnmxEETqTCecCpeT3dW1t	2.0794415416798357
PCN4IL8Sl7aemdILJ1dNh8VmAdr6x1	2.0794415416798357
lW30ngNcqL6mLiMRRDsF28OsTtQxTQ	2.0794415416798357
GoAv3IXUTekhiAQOgSCcU2OWBWANND	2.0794415416798357
tlT19bqp99iJXSDruT9LqVSFVvsLvQ	2.0794415416798357
SXToreP7mS4mjcrw1uCj0HViutjqmC	2.0794415416798357
w46pDADVMEEITLk53x6BRl1AHJM5XU	2.0794415416798357
kUmGcbJddc4rBVTlTTMqxSIReolkvv	2.0794415416798357
1pseeb3R8E4hwpGag2IPJkpW0u2DS4	2.0794415416798357
Gajk8ugfpiUcDCDN6lnat3RK1Lnkla	2.079441541679836
U5w3KaFRkK99WFLb4XRgfmUQUhlsUw	2.11021320034659
dQWJ2S94CSBxmMO9fqJxMdjIwdnEJG	2.11021320034659
l7xboC6Hu76p5D4Jo2rvHPS4gsPMTH	2.11021320034659
E3N74naULTD2G1PlpiqxopBwXPnEPM	2.120263536200091

```
ohq16LTNxpw36VRJr4N0ttKcGE4HdB       2.120263536200091
FMdP6don03udLbQAQiPnNo2abhGFeM   2.128231705849268
Vk2doIBUMEKU3vVrbO3k9uDEhvsISL     2.1400661634962708
x9MDjm0BRRf4fSa9skom4QhXcqiDls       2.1400661634962708
8neBI4519203PudofJkaGWKjw9o9T3        2.1400661634962708
92xq1wEUNphQe8dEDlTpJ2laRSo5K6      2.1400661634962708
9902DpKFAlQgbaV4KG090Bj9NTXjB1     2.1400661634962708
2ILT8sUXwrHA3BxINf9N1po8o2216d       2.1400661634962708
dvLmOqws1pIJ7IvqwSOCv04QNiHftd       2.1400661634962708
RoMqfrFg6a7Grj18EU3UqmbJ0RJOgo       2.1400661634962708
xmFwSDLCquMnaC2m2CwT8waM1uRSmo           2.1400661634962708
hSNF8n5dgldQfKSRfGqRHDWeWg5luV   2.1546649629174235
8INx50MexumO6DkJFESTcGs3QQW3pS  2.159484249353372
6u7oQe0K0E4oKvjBShkbktpgrCggWl        2.174751721484161
EFuNH2tJwpeONN1XAjsDxvaCIdchuI       2.174751721484161
dNbNjB6SgjAWffh5sVkX0OE4aGvU2B    2.182298927119544
TVmW8JfBgOxGEmLpJh5UWdApNurlLU 2.197224577336219
5R6sgaxLLW9KaXEMEBecQOruo4BchD   2.1972245773362196
0gvWD0vDouqfVsARqot7jMjK3VXHuL    2.1972245773362196
9cJwSRdGn4UrtmhqlIAfbvS72cDBDI       2.1972245773362196
VLt4V0Lu2sBuxJIMFbxPlT33Iq4K6I        2.1972245773362196
Time taken: 15.632 seconds, Fetched: 100 row(s)
```

# The result is advertisements ordered by a heuristic estimate of the chance of a click. Observing predominance of advertisements for Apple products.

# Summary

Developed a job flow to process impression and click logs uploaded to S3 by web server machines.
The result of this job flow is a table in Amazon S3 that is used to develop and test a model for contextual advertising.
The Hive statements collected are used within a job flow to generate a model file.
Uploaded the file to Amazon S3 and thus making it available to adserver machines to serve ads contextually.