CSCI 720, Section 03

Report for Project 3: Analyzing Large Log Files

Group Members:

1. Abhishek Jaitley
2. Palash Gandhi
3. Mansa Pabbaraju

Synopsis:

Big data techniques can be utilized to solve issues related to handling of huge data occurring in many real-time applications. The aim of our project is to utilize a "big data" technique to design and implement such applications which require a huge amount of data but have limited resources. For this project, we design and implement a solution for finding the 15 most occurring IP addresses in a corpus of web log files. Along with the 15 most occurring IP addresses, we also display the accounts accessed by these 15 IP addresses. Accounts accessed multiple times by the same IP address are displayed only once.

We have used the technique of MapReduce to solve the problem of finding the 15 most occurring IP addresses. Our solution will work for any number of mappers or reducers. We have perform post-processing once we obtain the results from all the reducers. For this project, we have used a simulated environment provided [1] to implement the MR (MapReduce) solution.

Reason we used MapReduce:

The input sample data are three large web log files. For these three large files, we have about 7.1 million instances[6]. Handling such large data is impossible for standard single unit processors. Thus, we need to make use of a technique which can 'divide' such a huge data file into smaller units and distribute it over multiple processors so that they become practical to work with. MapReduce technique is a good fit for these issues as it enables data to be divided into chunks and distribute it over a network of processors[4].

Also, the amount of input data is not pre-defined. Hence, our solution needs to work for any amount of data. When the amount of input data increases, we can simply increase the number of mappers and reducers to handle the same. Thus, Map-Reduce technique seemed a good choice for this project.

Analysis of input data:

The corpus of documents contain web access logs. Each line of the log file follows a standard format showing the IP address used for access, its timestamp, followed by the URL accessed. These URL's are either accessed by people having accounts on those systems or are other direct accesses to the URL. Each access, when involving an account has a special character '~' before the name of the account accessed. Others directly start with '/' character. We use the concept of 'Regular Expressions on strings' to parse the information we require from the web log files.

Preparation of input data:

For our simulated MapReduce environment, we prepare the data in the mapper phase which has its input as a string of file directory containing log files. For every line of the web log access in the input corpus, we read the line as a string. We further perform operations on this string to obtain the IP address used for access and the account the IP address accessed for the particular instance under consideration.

Obtaining IP address:

We convert the string representation of entire access instance into an array of strings by splitting on the space character. The first element of our string array is the IP address for this particular access instance.

Obtaining accessed account name:

For obtaining account names, we take advantage of the given fact that a special character '~' always appears before every account name of web log access instance. First, we convert the string representation of entire access instance into an array of strings by splitting on the '/' character. We observed that for every instance of web log access, the account name which is preceded by the '~' character always occurs as the fourth element of our string array when split on '/'. Thus, we take the fourth element of this array as our account name

However, there is an edge case we need to take care of. In addition to accesses involving accounts, there are other accesses also present. These accesses do not have the '~' character before them. Thus, if there is no '~' character present at the beginning of the fourth element of the string array, we ignore the entire instance of web log access.

Concept of our solution:

There were two approaches which we considered for this project. They are as follows:

First Approach:

The first approach we thought of involved pre-processing, MR phase and post-processing. Initially, we decided to use a data structure to store information of IP addresses and the corresponding accounts names which were accessed by them in the pre-processing phase. In the Mapper phase, the mapper will emit an IP address along with its corresponding account accessed In the reducer phase, the reducer would count the total number of times each IP address is accessed and emit the result as an integer in string format. In the post-processing phase, we would sort the results obtained from all the reducers in our MR paradigm and then obtain the top 15 occurring IP addresses. Now, for these top 15 occurring IP addresses, we would find the corresponding unique account names stored in the data structure in pre-processing phase and then display the results.

However, the main problem with this approach is that we need to read all the log files twice - once in the pre-processing phase and then in the mapper phase while we perform intermediate emits. Hence, the time complexity for this approach would be very high.

Approach we selected:

Unlike the first approach, we wanted to access the log files only once as the size of these files is very large. This helps us to save on time complexity. This approach takes advantage of the fact that once we read the files, we already have both the required pieces of information which is the IP address and the account accessed.

We read every line of the log file as a string in our mapper phase and emit the IP address and account accessed as a key-value pair. The shuffle and sort process aggregates this information and gives the reducer an input which consists of an IP address and a list of accounts associated with it.

In the reducer phase, for every input consisting of IP address and its corresponding list of account names accessed, we concatenate strings account accessed into a single string. While concatenating, we also add space character after every account name which we can use as a 'delimiter' in our post-processing phase. We emit this concatenated string along with the IP address value as output of the reducer phase.

In the post-processing phase, we first obtain the results obtained from all reducers. We then count occurrences for each IP address. The total number of occurrences of each IP address is given by the size of its corresponding accessed accounts list in the results obtained once all reducers finish their work. We sort these occurrences to find the top 15. For these top 15 IP addresses, we display the accessed accounts which are also available in the results obtained from the reducer phase. Multiple occurrences of same accounts accessed are displayed only once.

As the second approach has better time complexity, we decided implement the solution for this problem using the second approach.

Implementation details:

We have implemented our solution in JAVA called 'MapRedClient.java' using the simulator 'MapReduce.class' provided by Dr. Trudy Howles[1].

For the simulated MR environment, we assume that once we invoke the execute() method, all mappers are assigned their respective work. We also assume that once the execute() method returns back, all reducers have finished their work and store their results in the 'results' Tree Map data structure of the 'MapReduce.class'. Thereafter, we perform post-processing operations on the results obtained once all reducers finish their work.

Mapper:

mapper(): For every instance of web log access, we call the 'emit_intermediate()' method from 'MapReduce.class' with key as 'IP address' and value as 'account name'. Both key and value are strings. Thus, essentially, we 'map' all accounts accessed by a single IP address to that IP address in the mapper phase and emit it.

Reducer:

The reducer takes as input an IP address and its corresponding list of account names accessed. Both the IP address and its corresponding list of account names are strings. For every incoming IP address, we concatenate all contents of its corresponding linked list of strings into a single string along with a space character as a delimiter.

We call emit() method of 'MapReduce.class' with parameters as IP address and its corresponding concatenated string. Thus, in the reducer phase, we are aggregating all account names associated with an IP address for the current reducer into a single string and then emitting the result.

Post - processing:

In our post-processing phase, we use the method getResults() from 'MapReduce.class' to find the number of occurrences of each IP address and their corresponding accessed account names. Here, we use two Tree Map data structures, available in the "java.util.TreeMap" JAVA package[8]. We use the first one for counting of IP address occurrences and second for finding all the unique account names associated these IP addresses. We sort the counts of occurrences of each IP address in descending order

Once we find the top 15 occurring IP addresses, we output these IP addresses along with their associated accessed account names. The second data structure used is a Tree Map having key as 'IP address' and value as a 'HashSet' enabling us to avoid repeats in account names. We use 'HashSet' from the java.util.HashSet package[9].

Efficiency of our approach:

Tine: In the mapper() method, we scan the entire data once to read the contents of the log file. For every line of the log file, we call emit_intermediate() with IP address as key and account name as value. Thus, complexity of this step is O(n) where n is the number of lines representing single web access instances of all log files combined.

In the reducer phase we form a concatenated string of all account names associated with an IP address and send this to emit() method which outputs a key-value pair. Here, the key is IP address and value is the concatenated string of account names. This we do for every unique IP address. Let the number of unique IP addresses be 'k'. Thus, time complexity for this step is O(k), where $k \leq n$.

In the post-processing phase, we use local JAVA data structures to count IP address occurrences and the unique accounts associated with these IP addresses. Thus, the complexity for this is O(k).
Combining all steps above, total complexity for our solution is O(n) which is Linear time.

Space: For our solution, in addition to the Tree Map data structures uses in 'MapReduce.class' [1] to implement the Map-Reduce paradigm, we have used two additional Tree Map data structures for our post-processing process. Each of these Tree Map data structures has only one element per IP address. The values in our Tree Map are linked lists of at the most 'n' nodes where nodes represent corresponding account names. Let us suppose that there are 'k' unique IP addresses in our web log files where $k \leq n$. Thus, the space required for our solution is bounded by $O(4 \times n) \approx O(n)$ which is linear space.

Tradeoffs involved in our approach:

The post-processing step uses two data structures which adds to the space complexity of our solution. Memory is an expensive resource and data is ever-increasing. Hence, even with multiple mappers and reducers used in a distributed environment, space can still be an issue.

One more trade off of our approach is that as the format of our data files change, we need to change our data preparation process and the 'regex' operations involved. Our solution will only work for web logs which have the provided form of input data.

Evaluation and Accuracy of our results:

To test our solution for this project, we wrote a separate code using JAVA data structures to solve the problem and compare with the results of our MapReduce approach. The top 15 IP occurrences matched with the results of our JAVA code written to test our MapReduce solution.

Time complexity of our solution for the given log files turned out to be good with our solution taking under 2 minutes to complete execution.

Instructions to run the solution:

We expect the log files to be present in a single folder and the path of the folder to be passed as command line argument to our MapReduce client 'MapRedClient.java'.

There are two files: The provided 'MapReduce.class' file[1] and the solution we have implemented which is 'MapRedClient.java'. These two files should be in the same folder while running the code.

The code should be run as follows:

1. Compile 'MapRedClient.java' file using the following command:
```
javac MapRedClient.java
```

2. Run the solution as follows:

```
java MapRedClient <folder_path>
```
For example: java MapRedClient.java /usr/local/pub/large_log_files
Here, large_log_files is the folder which contains the log files

Output:

Please refer Appendix A for the output.

Conclusions:

We could successfully solve the problem of finding top 15 occurring IP addresses and display their corresponding accessed account names using the MapReduce approach. Our approach needs only one parsing of the entire input data files. We extract the required information from input log files and provide it as input to methods in our MapReduce paradigm.

We count the occurrences of IP addresses at the end of execute() method and not in the reducers itself. Thus, our solution does not assume a particular number of mappers and reducers and takes care of the fact that there can be multiple number of mappers and reducers. Hence, our solution follows the MapReduce paradigm and will work for any number of mappers and reducers. Also, our solutions shows a good time complexity.

Looking back, we think we paid more attention to extract required information. At first we decided to include an additional pre-processing step. However, we think we could have saved time if we would have considered the fact that input data files for this problem are very large and any extra pre-processing would be an added time-consuming task. For our final solution we have removed the pre-processing step.

We think MapReduce remains the best choice for this project as the input data was very large. For a real-time application, log files would be very big in size and it would be impossible for any single processor to work with such large data files. Thus, it would be necessary to divide and distribute the work into smaller units enabling a distributed environment. A MapReduce paradigm is a very good choice to implement the same.

If asked to work on a project with similar data in the future, we would base our solution on MapReduce itself. However, we would like to explore ways to save space as even when we work on a distributed environment consisting of multiple mappers and reducers, as we learnt that when working with a MapReduce paradigm involving huge data files, a minimal but sufficient approach for pre and post processing works best.

One of the ideas we can think of is the use of 'bits' to store information and do some part of processing involving bits rather than memory in terms of bytes. This is based on the concept of 'bloom filter'[10]. We would try to explore if we can use such logic as either part of our MapReduce paradigm or in pre/post processing.

References:

1. Map Reduce Simulator by Dr. Trudy Howles

https://www.cs.rit.edu/~tmh/courses/720-2016/MapReduce.class

2. Reference for Map Reduce simulator document: Dr. Trudy Howles,

https://www.cs.rit.edu/~tmh/courses/720-2016/MR-Javadoc/

3. Slides on Map Reduce Pre and Post Processing, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/ MR-PrePostProcessing.pdf

4. Map Reduce concept reference: . Slides on Map Reduce Pre and Post Processing, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/BigData-2016.pdf

5. Project Report Format credit : Dr Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Projects/Project 3 Report Template

6. Dr Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Projects/Project3-AccessLogsWriteup

7. Input data reference: /usr/local/pub/large_log_files

8. Reference for JAVA Tree Map data structure: http://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html

9. Reference for JAVA Hash Set data structure:

https://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html

10. Slides on Bloom Filter, by Dr. Trudy Howles, from RIT mycourses/CSCI.720.03 - Big Data Analytics /Materials/Streams-2016.pdf

APPENDIX A.

Output for the given three files 'access_log', 'access_log_3', 'access_log_new':

```
#1 occurring address:  129.21.30.99
```

```
     sns, icss485, sfj, rdn, gameai, mbp, phl, icss361, jdb,
     rlc, jlk, mjf, mjc, jdg, bdm, jde, mjh, ncs, txb, sew,
     may, ero, icss234, rkr, icss233, icss232, wmm, icss231,
     icss352, chanbox, ark, hpb, icss235, dgs, eal, temp361,
     dgt, jch, icss351, gdpro, aag, dgj, rsg, ritra, wmc,
     pga, pxseec, bks, cmh, tfp, sevolume, mlw, ljt, shk,
     sps, aprese, bga, tjr, icss544, emc, se420grd, spm,
     nff, icss263, spr, csgradco, jfn, tba, sxn, drw, paw,
     lrr, chr, mdd, sph, sxp, jfg, ats, se362, se361,
     solubox, mulmedia, sxj, ell, drp, atk, rep, tah, atd,
     nmp, se-admin, lac, se442grd, anhinga, kxa, mks, swm,
     kgb, dad, tjh8300, hae, jeh, csdept, probbox, patric,
     jmg, icsg720, kjw, sjp, rpv, fyj, mwa, ewo, lr, ptt,
     vxs, labman, aob, wrc, it-fac, waw, rpj, kzp, sal, jxs,
     bhh, mn, cbb, kjb, eh, csdoc, fxk, kar, afb, vcss345,
     se420, cit, TTP, leone, ngm, mmr, mek, amt, vm, ems,
     kaa, cs0, cs1, hkd, rwd, amp, cs4, gig, icss707, cs2,
     cii, cs3, netsyslb, icss221, ns, icss341, lwh, jrv,
     newsmod, mpv, epr, nrd, bjk, gj, dnc, jam, wcc, se441,
     se440, icss456, icss334, icss455, scj, se442, tmh,
     grad, icss450, icss571, icss570, jal, csx, tdw, cslab,
     axa, jaa, jab, wjs

#2 occurring address:  202.64.111.130
     ats

#3 occurring address:  108.45.93.85
     ark

#4 occurring address:  129.21.36.110
     displays

#5 occurring address:  129.21.36.111
     displays

#6 occurring address:  68.180.229.52
     lr, ptt, vcss231, vcss232, labman, vcss233, rlc, jdb,
     wrc, royale, grd-800, waw, rpj, ib, mjh, bdm, ncs, eh,
     icss352, csci101, kar, csci140, kxb3682, afb, rmp3621,
     hpb, ark, csci142, anh, icss235, dpk3062, grapecluster,
     TTP, icpc, icss351, mmr, amt, axr1545, apr7264, pga,
     rwd, dnc6813, cs4, arl9577, rmm7272, vcss707, vcss23x,
     sps, csci453, jrv, adb3160, gxw9834, spr, mpv, zjb,
     sxn, paw, rlaz, chr, rjb, grd-665, ats, csci603, dprl,
     mtf, dmrg, vcss243, tmh, gs, anhinga, icss571, swm,
     vcsg720, dfr2936, jeh, pf, jaa, jmg, icsg720

#7 occurring address:  66.249.67.185
     ark

#8 occurring address:  66.249.67.206
     kar, ark

#9 occurring address:  66.249.67.198
     csci142, ark

#10 occurring address: 129.21.37.155
     hpb

#11 occurring address: 211.216.103.188
     sap1646, kge7490, exl9109, mxc2967, bwh9324, jxl8273,
     axk2789, nrg1142, phm0150, bff3933, rlc, cmf5736,
     dnw0171, jcr2645, bws4435, mdb4562, ncs, amw8338,
     sab0343, asb9949, jej4130, brg6507, p719-01c, ayg9566,
     dsm5856, alc6276, p719-01a, p719-01b, rkr, mjk7909,
     dmd9910, dih0658, cdr5441, jls9988, sxl6484, rxr2990,
     nes7983, acy7841, pga, anm0944, dab9353, jak7775,
     kxh6049, jjc1752, txz0309, aam4510, brs1873, cxm1147,
     drh3352, ajh0995, mbr8040, ece2681, hjg7225, eds2064,
     mdd9409, bjb5557, axv6446, jmm0257, dar4731, kha5670,
     ses6442, jdc9088, jja7656, wpf6749, rtk2978, jag8153,
     bjr9531, crm6425, mss9221, ssa0821, rro0366, wfm8778,
     jrv4064, jrk3666, dpp3801, lac, bwl9817, cs1-grd,
     jaw9088, cmj8459, pkn1587, rtc0400, bkj0604, acp0238,
     pmj0460, aph7562, rsc7461, mxd4716, nxg6191, rms2456,
     cxv1089, fxm1050, srp9633, exb6767, asv4971, adl8790,
     sja4756, jxs8185, jmp9746, nxk5767, esr4485, wjb1253,
     cxa0702, jcs5924, lgp8996, mrt6315, erd4819, mtp6152,
     rpj, mgb4668, gxv0796, ajr5788, src1504, eaf3095,
     ewr6582, bcg1215, mrs4108, gdh8512, bjr1822, txl2715,
     sfg2589, mcd5210, TTP, adw6302, bpm9413, psm3971,
     bul0707, amt, apr7264, drv3988, pxs0030, cs1, tad8375,
     cs4, txl7166, cs2, sai6189, cs3, bmh3937, dei1809,
     frt0714, rwa2233, avm7370, rdw4659, ens9125, wxa1644,
     djt1765, mws2969, msd6123, afr6941, cas0921, xyg3744,
     jaa4323, wtm2326, gwj3667, jeb4632, alg9820, cfc9018,
     axm4991, rxb1824, agp5411, jas3900, cs2-grd, gah6035,
     zly1308, jrm0182, ntt5186, mja0336, jmc4295, mxh8303,
     ixs4381, zcf1470, ask1725, axj7140, bjm6162, jxg3776,
```

csx, aks9587, mjs4379, kxb5919, dxk3582, lxk6558,
mjv2816, axt8828, jaa, dmm5208, tmn3144, tis0633,
yxl1461, djb7630, ads3823, jrp9628, sda6161, prm2412,
mxo5559, bec2192, bbk8348, sxg7909, mjw5708, jkc9915,
pak1829, jbs8310, rjr7401, cxd5608, brb2956, ofp8263,
aae9439, dbk5932, svc5135, pvf6609, ksl7480, jhb3827,
erc8572, jdb, jbc7075, nmm1099, afk8836, rtm9160,
bsh8976, pmf4570, hxp8813, jsr7584, csd4312, ark,
saw3763, shd0326, yhy6841, ncw5953, mal2440, rsg,
amm4053, ejh0503, pwk5869, jzs9783, ajb1391, kdm8123,
rxg1272, maa2454, mwm4201, dhv5289, agw0079, jpk5521,
tjl7666, jib9640, txs3788, trs9763, mpd5171, cdg2433,
dvg8277, nxg3751, rfm4792, ats, caf6283, ass0373,
mar0346, lwi3690, atk, smc1342, zxa5730, rcb9104,
ajm1072, jwf0207, mrt5416, vah8906, dab9721, jhg5003,
tml2825, sjg2490, jdr3760, nxh7918, hae, dak2255,
bes1251, jeh, ajk8714, jxp6713, jgs3280, rlc5227,
atg2335, nmg1829, dms5732, dmb3357, spm4006, cxs0778,
dws7419, rsh7902, ljc0121, amt6788, ptt, mxb5157,
rdb6967, fws5777, vxp2888, was7560, ext3994, kav1180,
bph8876, waw, dpk0854, tdw4374, rpk4420, rwt1008, csx-
grd, sab1637, jrc7976, nyj4905, mda2376, aes2336,
sls9925, asg8525, clw9939, jxy3580, jjg7797, bxc2495,
lbl6598, aeg6147, paj9563, sks9267, avg1977, awo3207,
kav3351, cxv6719, trc2876, mab0270, sws6085, jcb1541,
rwd, jmm1192, jmj2413, dvc3336, dck2226, icss707,
ndm5017, dal8623, ajr3508, kcr7135, sxs5438, cdm0312,
rds2653, tjw1881, mws1470, ajd5356, aec2039, jmt3447,
krm8741, bxg6007, seh2117, psn8138, rac0595, bpp7700,
fhd6554, str8887, acm0810, twm6716, jdb5575, mez8702,
jms2551, jmw9845, sjb6353, dlg8894, mas9145, yrl9224,
akn0473, crh3333, brq1161, cslab, rwb4850, asm6855,
mxg5630, sab1867, bjt3697, bje8654, axt9690, aan0736,
saj5020, mrd8277, jmw8161, dxo3485, krm4686, glr7495,
pcf2159, tas5509, mxh0011, amm1864, mas8820, jnk8637,
jcl9307, ats3085, res6571, twm6643, njd0382, tlg7792,
jrl8103, icss234, icss233, icss232, rjh4581, icss231,
xxg7898, coh6585, jwg8832, wxt2966, tad4649, icss235,
djk9149, iar6219, pns6910, tjd6352, rgr8261, msm1074,
rls6543, blf0582, rds3792, mpw0220, jwm6981, dcr6918,
rmm2948, trs7058, jld0484, kfg5745, cad3658, mgs4954,
jaa7104, jmr1287, mpl0907, pfw8521, hxz9703, omc4527,
jpp7048, dcd3510, prr1313, csg5684, icss263, jdh5469,
nfh7354, ads4324, bjd2207, jem5451, mjw5652, jmf6689,
axj7737, tml7562, knr8293, pcb1932, lxz0062, jwl2319,

bgt0823, tvp8500, dmt0002, jrk5050, cxc8926, jsf7949,
jmp2961, jmg, dfm4737, djm2370, bss7325, gtm4972,
jms3453, ntm8059, tak3850, jmb2618, wcc2987, msw6017,
jdw5278, mxr4326, mhn5054, jbp3930, dbd9785, jxy3518,
ajd5295, smh7237, mxi2027, nxy5902, ljm4003, apl0823,
dij4361, shk8974, imb8999, rxd7870, axs0401, clh5014,
nhb3306, mjs3814, mtl4490, cef2163, jpm8738, cxw2229,
ear1856, ark7289, ang6829, esd8245, kkm2002, tpr0350,
mds0111, jjh3710, kxm6065, scs5637, rma4738, pme2880,
bgk3313, gxc3921, als1140, icss221, adr7826, sbp8988,
231-grd, cxc2192, jrv, yyl8077, jca1234, srb0036,
kmj9907, cdc2771, jdh8946, stk9297, wem8247, jdf3402,
cmr9845, smj1327, jmt8734, djd6828, etk1202, dpl0806,
pxc9821, pjv6136, tvn6705, plw4632, slg2241, icss456,
dab5849, bum6876, kjn8626, jmp8060, lcd3279, ajq7237,
rtm8476, gad6055, jwf1101, cmw3741, cll4352, cdm3791,
pmv5693, aat0995, dlb4430, cas8092, swz1316, jwg9538,
aeb2731, aaf6525, rds4018, pjl6624, eck2924, ajg8850,
cssac, khl0609, whh5154, wen2559, ach7352, mgn8400,
tim9108, bdm, mjm9787, mcl4080, mkm0466, cjs5459,
amj4510, mcl8699, hpb, jbg7763, nwb9412, lad6018,
lsh6478, wxy1291, smb8759, smb9848, mkw0550, jpc7937,
wes1735, mnc2986, prd9104, jmw5506, mjs1445, sds6656,
msk7188, mmb5486, dpm7203, des5122, njs9152, smt8716,
man7087, ifc1592, zpl8666, cxt3276, dmi0772, mwm8006,
adm9740, kms7407, pjg0606, yxl2809, bmm9186, kse9532,
dac3892, mjs4983, kcl8605, jfb7079, jah9814, jas4237,
aee3816, sxl0436, jpm8097, pmm1626, sab3317, axs3497,
awe9531, sxn1122, nxs5608, cbh5841, alp0091, jaw6099,
gdg1956, sem7610, gkm3997, anhinga, pxg5240, mxl6896,
rdb9723, dxm8021, jat5375, cxe0639, 232-grd, arr1681,
jxs2305, mrs6264, jwe5789, cbe9582, jjm7570, kmm7169,
dag3259, gdk1300, ash0151, tcr9166, owk5175, ktl9989,
bsg3433, mjv5268, jmm1885, kjw9644, wrc, gck7129,
txk0229, jgr0346, sbs1947, jnt1482, rck6844, rlr6379,
bmh8691, ffg3183, mpn8095, boa8488, jkg2154, rce9873,
jrm2428, sjo0940, vxs5163, srb9954, sxm4866, ajd0690,
cgz3317, dcc1977, srf2754, sas2453, wcr6041, mds6058,
jak5034, bjc9019, vxd4504, brg0853, kbc4660, gvs4271,
spj4251, hhh3016, tmb3006, mkp8819, jln7449, tar2893,
bkm8836, ttn6213, mxw6550, jdr7376, dws9263, jjr3455,
cab9395, mdl8158, jim5870, mek6721, jam1758, shm5473,
brs1429, sak6988, swt5482, mdm3513, aac5839, ajk8244,
jbs3722, yfz1745, rjv6512, jrs8685, jdh3090, ken6134,
hwl2309, cxz2509, mbd0240, dab6189, yxz9279, lbd7466,

jmb8755, eka0998, jfg3119, kdh7962, abg4155, mpv9903,
kjj5745, prd6375, rwt3740, bpc5860, cxp7449, gxr3286,
mck5419, cxm3417, bpw9015, may, mpm6910, ath7325,
jma8721, ixo7295, kgg2775, ams2449, sxg0562, cjh9783,
dnd9403, pdr8125, mja9444, rpw7781, jjm5748, jxs1878,
lfm8184, cxo6309, sxj6976, mjl7615, prm2951, lxc5005,
mcw4909, sxt0706, bjt0654, mlp3370, fpc1883, rac1915,
ajb0329, beb5913, ssa9529, adb6790, spr, std3246,
pss2417, lcy0304, jrs6735, sxc5568, mrt6695, ajo0548,
sng5511, smd6304, swh9110, mgp2775, lrm7820, axs0148,
cxm1497, sam2195, kas3897, jrc0362, wph7530, jsk4445,
jms0218, eab3572, jts0422, axa6017, gfc7581, jxr6022,
ajo3809, jag8260, csdept, ljb6563, jlf1748, jmb5267,
svm4777, rxk4983, jsb1220, icsg720, mjr2016, bsc2607,
fyj, cxp9817, fxl3374, pjn7364, mjl4772, csh3847,
jaj3213, slr2777, jpm4819, jfm8918, kcs1932, emt2435,
dxf5187, gsp8334, nch1255, djs9304, atm4339, cpl9150,
bcm6093, kjt8342, jat9360, ura1096, cxl6084, jst3290,
kar, jrb3608, all1205, scl7094, lmb8845, cdp3511,
smb2543, cks8804, dij0573, kaa, tak9830, kdh7931,
tjm5347, dlr1225, pgk7025, aao4436, djr7581, jtr0849,
jxj8963, jkg9634, war8797, sxd8607, jjm7995, sxl5090,
bjd8112, jsg1324, wmg3051, cen0282, exr3177, prg1500,
djb2070, rjs3356, gik2044, pmt6263, mdw4361, txr0310,
ajn9108, njs8030, dpg2099, awh4992, krh1124, jsr3204,
mcd2281, cmr0348, cxa1949, cmb5199, tmk4722, jrd6974,
gxy1388, lws7636, bnr3839, dsd1407, agp7723, jbg8713,
dst1011, sxj3407, tjt3473, cma8660, sks0995, nxp6875,
sml9350, jhl2141, cxp2822, dxy6794, jac3787, sgu4018,
mjl, rwm5894, dxr3738, jiv4926, jdl8219, twl1797,
ajd4166, dac6384, ers3268, mac4118, whb8352, tvo9075,
mpr9284, sef7055, mxs9917, ejk6825, fxs0344, dnh5577,
ghb1001, led0041, dis5149, tdp4357, smp7886, dkk4399,
kas6073, ask9157, jds6925, jeh5581, dnn0182, lxe2883,
mjh9979, twc3796, cxm7408, bmb2063, snk9934, reb9947,
softeng, stb0565, mdg9542, jst1734, jrms, nwm6357, sxn,
cxl7607, mcr5149, adb0840, pxv3951, sja5146, jab0016,
bja4700, afw0359, jah4926, atc5067, mxj1816, pfk6942,
rtp4264, pmc8184, mrb6858, cxk2598, scl9891, pso7025,
wck1910, jhk6964, swm, deg0873, kml4510, pac5854,
cww5940, thl5901, idw0680, sac8371, dbw4795, mrh1825,
msr2483, tfs8152, jar6070, smb8062, svg1390, dgg4028,
wds0313, jmd8720, jal0193, mdp1758, pxs5461, hha2316,
jmk2573, ajk5554, ajm3144, tjw0890, jsl8286, dtk2037,
cxv6847, ege4864, rbm6563, mxj2084, axj0483, rml1343,

kmc6820, spy5035, cac8319, mai2721, rcb0622, bjc6310,
byl1828, eh, csdoc, jar3817, jmf5071, emk3594, dph1174,
exc5972, mmr, bjh9235, fds4352, krs1576, ajz0465,
jjv2550, jsb7384, mej8766, mad3251, dsm2431, ams6686,
bas6542, irr1449, taf8925, jda4552, ner9848, rsc4493,
rat7079, aej2574, wjf9385, kwt0195, jrp1636, jjp6672,
pbk6628, sjy1244, amb9845, jmm5433, pcontest, grd8301,
hfd9603, eer7286, mpv, ajk3594, jaw3289, pmw4328,
tpo7551, esb9138, rjl7002, scd1879, ndn7322, dem5302,
rcs0940, aep1966, jam6764, meb0984, rkn4233, vxv8403,
jdb1090, cjr2885, mjk8256, hlm0011, ksb8327, tdb5818,
jew4884, jmj0373, rlr5874, mek3361, cko7622, cpb2511,
pmm9287, mkj7162, clb1515, jxk1035, jmm3077, mjr7469,
rmp4664, apm7109, rtb1145, cxb0025, smm4301, icss361,
gjb9156, cxa9275, ejj5875, mps6427, dan0337, nxs8185,
arb8320, kcb6278, djh6585, vms3120, dmm0482, cxg1624,
kds6767, mam8624, mdf9907, icss352, sxl1235, stm0164,
sle2954, pdj1133, brs6415, jpm7724, dej9962, nft4678,
icss351, jct9345, jpa8684, gar8595, mjd6285, gca9147,
axm0882, mhr8093, rnb1914, mjl1517, jxf7315, cxc3121,
rjh3141, bah0069, leb4189, mbm7297, dzz6876, glg5760,
stm1461, tcs8379, dmf2783, kem9593, keb4779, jmp7064,
apr6775, syp1865, lam3434, kwb1261, gxo8505, mwc3627,
rss5553, kjr8111, sjc4866, dcw7369, wjp5442, mcg3481,
mvl2421, jcc9715, mlb2536, jsl9020, jlb5055, bjs0068,
ajk9893, cxw0106, djr1572, jwr0593, pmh7814, dxp8856,
eeh8894, jak1456, sxj7122, mjd6022, mrh2075, sap6210,
hbl7107, mpp8195, dfm4613, maa9228, rks3877, bmg7879,
nxb0874, arl2562, srg8317, djl2401, vxk2549, mjd9351,
mgg4494, slp2009, akc3510, jxk6924, bws0794, tan2056,
mpm7481, edh3168, jxd0049, mdr9016, emd2175, ajc2926,
sxa0604, pxz3838, yxl7594, mpb6705, axk4132, jma8000,
aar4623, ial8431, rog6556, rxs4164, mvl4665, anh3243,
rwk6520, mkw1714, ebk3521, cxl7616, jat2166, sam3814,
clq2535, smw5352, emd8928, zes2712, mwm8326, icss341,
dcr4426, mrc8531, axk6526, smc7112, brj6969, rmb2682,
jam4981, mmm6350, djh6367, nlv1471, ewy1521, cxd6230,
slr8624, pxc7761, kas8587, 964www, cxp9009, sxa1919,
sth0719, eam8920, raa1431, rhs7867, vyk5113, kxb4032,
dbl3791, icss334, bms6921, cxf0517, eaw5103, two6384,
icss571, spm3529, tmw0786, jmr6838, amk9458, ndk4275,
jfc5838, nll8189, maf8113, jmc6907, jrl9768, cxc3158,
nye1124, jxd2255, jco9473, mdh5324, jam9171, cjg5883,
ghc9465, tgk3856, wjl3679, pjb5577, 341-grd, dmh7224,
bpf0658, pmd8870, sxg7108, ash9053, mbc3675, cmh9427,

```
     yxh6600, ksb5668, eim7101, ajr6671, axr4428, rgp5682,
     zmm1087, tsm9041, ram8009, cem9314, csg3319, bmp4235,
     rja1587, crj6092, cje3841, pxm5860, rns5045, kxk3045,
     drr1980, pdd6808, mrd1109, dlb7689, axr7927, kfn2883,
     mds1761, jxu8817, jls6515, jhr5854, rmw7783, jmh8525,
     mjh1697, slm5989, mmk8354, cll9707, jmm8340, dci1910,
     rhk8830, aprese, tjr, kmv9165, dvp2326, pjb6400,
     mje6229, als8770, kaa3496, ccm2451, bxs6404, pac6393,
     jek5522, nxl3484, paw, sag1552, mtb3379, bjs7527,
     amh8145, ber8532, khw5093, rrs0174, agg1259, cxa8815,
     pxl8094, rak8870, jrl1209, beb1964, cs3-grd, skg2236,
     sjl5824, jjf0114, apw9673, jcr0038, lea1292, cxl7279,
     mxm6574, mtj5248, nrl5157, oam5145, lpr8870, cxb0906,
     rrb5014, maz1136, kgs2652, drp4138, dxh1031, cmc5608,
     rmm0616, njs4614, bll6037, mjg1676, jww4304, cxb3178,
     ewc2599, jls5634, apm9733, zik8947, mxm3254, hxt0072,
     jfz5661, dgs4466, taz9290, jxp1724, jwp2095, afb,
     sdl2927, jht6586, mjs9080, msw4585, rph5670, jbo5112,
     kmw0567, sxc3971, vm, axr4217, nms6646, jxw5130,
     jba0125, jjz3641, djh4350, khk8873, jkp3250, pjw9720,
     lcb5106, bld7273, sfa5313, abc8518, prs5445, bwg0858,
     vjf6412, jkl3831, jxj5606, mai3116, dsc7420, kbn2060,
     svd5439, sjr1521, cfl1375, mjm3062, jxp2608, axn3732,
     rch1402, rzf2382, nrr8953, cer4356, mam8484, nmr2279,
     tmh, sls6076, txo8933, jrk7389, frb1784, jmt5172,
     bdw5370, asc6380, dfr2936, bcb2708, ril2829, twb8626,
     yxh6894, jma9506, crr9465, pcp0899, slv1481, lah5523,
     dwl2239, mxd9928, tcs8192, jpb8058

#12 occurring address: 216.39.48.141
     axt9690, jjg1764, dre9227, cxz2509, p590-73g, aam3089,
     jgs3215, sap1646, yxz9279, drk4633, rtb1145, icss361,
     rlc, ats3085, aab1217, ncs, rjw2183, may, rkr, icss234,
     icss233, rma4653, icss232, icss231, icss352, nxj0619,
     coh6585, wxt2966, icss235, bgs2187, icss351, jct9345,
     jpa8684, gar8595, pga, mcr1276, jxs1878, kxh6049,
     jxf7315, djs2575, sps, lso8219, glg5760, drp2179,
     mdd9409, spr, icss263, std3246, jle1028, wxc9861,
     axa2446, lrr, yxh0366, tam2161, urosan, wfm8778,
     kas3897, lac, ava2675, cs1-grd, cxw0106, mxp2436,
     tjh8300, gsk6703, csdept, rxd9507, nxg6191, jsf7949,
     jmg, yxt0376, , icsg720, fyj, qxs8665, lr, pjn7364,
     hhp9485, rpj, axj9577, mdr9016, jxy3518, gxv0796,
     mxi2027, nxy5902, fxk, aar4623, rdw8666, kar, bjr1822,
     vcss345, arv3501, mvl4665, TTP, rut8402, ang6829, amt,
     ebk3521, cks8804, kkm2002, jjh3710, kaa, cs1, sxm3011,
     tad8375, cs4, dlr1225, cs2, yxc7227, cs3, djr7581,
     zes2712, rmm7503, dvn7806, jxj8963, icss341, liam, 231-
     grd, sxd8607, mal0568, dcr4426, jjm7995, jrv, kmj9907,
     djo2717, bxs7266, cen0282, prg1500, tjz3155, gap8763,
     wzh7428, sxa1919, ewd5363, rsk2445, qxx3451, yxv4997,
     njs8030, vyk5113, dmrg, icss456, icss334, two6384,
     jxs187, icss571, rxd8065, csx, jmr6838, dxk3582,
     aat0995, jaa, axt8828, kls4584, maf8113, rxg8002,
     yxl1461, rgh3595, nye1124, mdh5324, p532-70a, jkc9915,
     pal6640, axk0088, p532-70c, p532-70b, 341-grd, mjl,
     pvf6609, bmm3056, ash9053, jdb, wen2559, ajd4166,
     jbc7075, mjh, bdm, mcl4080, mnc5689, cem9314, sef7055,
     arl8258, ~cjm5578, hxp8813, jsr7584, hpb, ark, jbg7763,
     dnh5577, dis5149, mrd1109, mws7323, rsg, mds1761,
     mjh1697, rxg1272, sdm6689, cxt3276, jdd7555, aprese,
     icss544, softeng, jst1734, txs3788, mpd5171, jrms, jfn,
     sxn, paw, dac3892, kcl8605, jah4007, jfb7079, ats,
     pxv3951, se362, rrs0174, xxy4991, atk, hxy8630,
     beb1964, anhinga, pxg5240, cs3-grd, sgd1380, swm, hae,
     jeh, kml4510, lea1292, sks1282, dmb3357, jjm7570,
     spm4006, ptt, mxb5157, dxh1031, gjn3855, rxh7017, wrc,
     vrp6084, waw, kmc6820, mai2721, rlr6379, eh, csdoc,
     jjs9804, gxs5626, sma2347, asg8525, jwp2095, nat0001,
     dpk7386, sxl2521, afb, jck7765, emk3594, vxs5163,
     msw4585, rph5670, mmr, jbo5112, mas8625, vm, jdd5747,
     jsb7384, rwd, irr1449, gradlife, fxs8914, bid5275,
     djh4350, jdb6098, lxs5032, mpv, abc8518, bxg6007,
     mai3116, sjr1521, jam1758, nrr8953, se440, mbp3753,
     tmh, sls6076, mjk8256, ksj1579, cslab, bvs4997, xxh4486

 #13 occurring address: 65.214.36.117
     bjd9660, cxz2509, mbd0240, dre9227, sap1646, ewm5184,
     jmk7027, rlc, dnw0171, ncs, p719-01c, ayg9566, may,
     mpm6910, p719-01a, gmm1616, p719-01b, rkr, btz6419,
     p759-06c, sxg0562, dnd9403, dih0658, jls9988, kgs7001,
     jxs1878, pga, lfm8184, cxo6309, cmh, kxh6049, hxg4229,
     jxt5909, jcs2350, jjc1752, lxc5005, aam4510, jmp9921,
     sps, cxm1147, ksh4250, mdd9409, drp2179, beb5913, spr,
     std3246, jmm0257, umagwww, jxs0991, yxg7752, jja7656,
     jdc9088, jlz6576, urosan, lrm7820, sam2195, kas3897,
     jsb1579, jrc0362, ava2675, lac, mct4270, tjh8300,
     gsk6703, emt3734, nxg6191, rms2456, icsg720, fyj, ewo,
     esr4485, jcs5924, p590-01d, jfm8918, erd4819, kcs1932,
     fdn2479, rpj, p590-01c, axj9577, rct5472, gxv0796, kar,
```

bjr1822, vcss345, crn7319, TTP, cag1258, bul0707, amt, jac4020, kaa, marks, cs1, tad8375, cs4, dlr1225, jrg0839, cs2, sai6189, yxc7227, cs3, djr7581, rmm7503, jxj8963, frt0714, cja2729, bxr5159, pes5750, jsg1324, krb8621, msd6123, cen0282, mdg5943, drg4688, bjs2716, wtm2326, rjs3356, gik2044, rbb9168, ewd5363, qxx3451, egr1630, njs8030, krh1124, rxd8065, axj7140, jxg3776, csx, jaa, gxy1388, yxl1461, rew8422, dst1011, wad2692, ase6246, axk0088, ecw6651, svc5135, bmm3056, p361-45b, jrf0605, jdb, wgh0532, mjh, dcc5500, rtm9160, sha5239, eae8264, yxy7691, sef7055, arl8258, hxp8813, jsr7584, jmw9666, ark, rww5745, ewu5268, saw3763, shd0326, jru1146, yhy6841, dis5149, led0041, pwm5696, ntv1359, rsg, rxg1272, maa2454, dhv5289, softeng, fbl4818, jst1734, jrms, jfn, dtv6770, sxn, sxm, mxm4723, rfm4792, ats, caf6283, se362, atk, rcb9104, txl0712, scl9891, p361-42a, gab5730, swm, mjl2788, nxh7918, jeh, sac8371, jdr2670, dms5732, lxg3306, jag6485, dms2252, 934www, pxs5461, ptt, jjb3925, asl1140, dtk2037, jsl8286, ifc, waw, axj0483, hxl9267, rpk4420, cac8319, spy5035, mai2721, eh, asg8525, nat0001, kwf9352, jmf5071, jsg3432, mmr, nos, bjl7507, etf2954, bjh9235, p544-01c, ajz0465, jsb7384, rwd, irr1449, dvc3336, meb8208, icss707, rat7079, kwt0195, cms2264, kjb0566, sxs5438, pcontest, mpv, ajd5356, pmw4328, tpo7551, bxg6007, p590-99b, scd1879, mez8702, se441, se440, mjr8305, yrl9224, cpo2571, mjk8256, cslab, bvs4997, p362-41d, jmj0373, p590-73i, axt9690, p590-73j, p590-73g, p590-73h, aam3089, 930www, jvl2957, p590-73a, jxk1035, p590-73b, glr7495, apm7109, rtb1145, p590-73e, p590-73c, p590-73d, jnk8637, rjw2183, icss234, mam8624, icss233, icss232, zxj5191, rjh4581, icss352, coh6585, stm0164, icss235, pdj1133, rgr8261, icss351, nds8785, gar8595, kfg5745, fsi2630, mjd6285, ams6008, cjb2259, cmb3548, mbm7297, prr1313, glg5760, icss263, lam3434, jfc4899, gxo8505, lrr, zfh7894, jxm9755, dsh8290, jab0655, efg6144, p570-01b, dcw7369, deh3895, jsl9020, tvp8500, cxw0106, mxp2436, dmt0002, prs3594, jak1456, jaf7936, ceq1364, exs6602, rxd9507, mpp8195, mgw6456, jsf7949, jmg, nxb0874, djm2370, jcp3377, lr, 940www, wcc2987, ags4764, mpd3564, brr7907, kxc8696, edh3168, mdr9016, f2y-grd, ajc2926, nxy5902, jlz5863, aar4623, rdw8666, mvl4665, jjm6072, rwk6520, ang6829, mgs0394, kxm6065, sxm3011, pfn4402, zes2712, icss221, cxc2192, jrv, kmj9907, kjd6081, dlt1947, tjz3155, gap8763,

```
     wzh7428, 964www, rsk2445, mpm1734, dmrg, icss456,
     icss334, dab5849, icss571, jes1789, rtm8476, amk9458,
     cdm3791, kls4584, sss6686, cas8092, smc0857, mdh5324,
     kmb0969, rds4018, cac2372, khl0609, wen2559, mgn8400,
     axr4428, cem9314, dlf9078, hpb, lad6018, ksm4328,
     mrd1109, jmw5506, wmc, mds1761, sds6656, jhr5854,
     jcb7565, pjb6400, jro3139, dab9159, paw, rjz7584,
     njs3983, axl9127, jah4007, pmm1626, nfc5382, xxy4991,
     jhk0504, pxl8094, rak8870, beb1964, anhinga, pxg5240,
     oem2374, sjl5824, tad5684, apw9673, jon2401, lea1292,
     mtj5248, kmm7169, ash0151, thb5845, dxh1031, jwf6545,
     dlm8689, bll6037, rwc2810, wrc, wjb9256, vrp6084,
     apm9733, aor6022, rlr6379, gma6188, adg1653, cxc5157,
     sma2347, boa8488, sxl2521, jwp2095, afb, rce9873,
     jht6586, mjs9080, rph5670, dcc1977, evc1822, vm,
     smd8797, jdd5747, pad7335, djb9751, brg0853, fxs8914,
     jjz3641, bid5275, cjs9870, dxb7413, hhh3016, tmb3006,
     mkp8819, lcb5106, abc8518, prs5445, mwd7087, afm2162,
     sml4242, btr5290, sjr1521, del7479, nrr8953, ded8495,
     tmh, mdm3513, sls6076, aac5839, gcw1418, kxy6184,
     jjt5695, crr9465, lah5523, jtm1748, xxh4486, jmb5063

#14 occurring address: 17.138.56.17
     ark

#15 occurring address: 216.88.158.142
     axt9690, cxz2509, jgs3215, drk4633, icss361, dnw0171,
     ats3085, ncs, may, rkr, icss234, icss233, icss232,
     icss231, icss352, icss235, bgs2187, icss351, pga,
     mcr1276, jxs1878, kxh6049, jxf7315, glg5760, drp2179,
     spr, icss263, std3246, jle1028, axa2446, yxh0366,
     ?????, sgd9494, tam2161, urosan, wfm8778, lac, ava2675,
     cxw0106, cs1-grd, mxp2436, tjh8300, gsk6703, csdept,
     nxg6191, jsf7949, jmg, icsg720, fyj, lr, pjn7364,
     job1017, kkm7942, p590-01b, rpj, axj9577, p590-01a,
     mdr9016, nxy5902, aar4623, rdw8666, jst3290, kar,
     bjr1822, vcss345, mvl4665, TTP, cag1258, ang6829,
     rut8402, amt, cs0, cs1, sxm3011, cs4, cs2, sai6189,
     yxc7227, cs3, djr7581, zes2712, jxj8963, icss221,
     dvn7806, icss341, 231-grd, liam, mal0568, jjm7995, jrv,
     wxa1644, kmj9907, bxs7266, gap8763, wzh7428, ewd5363,
     rsk2445, yxv4997, njs8030, vyk5113, kxb4032, icss334,
     icss571, jmr6838, csx, dxk3582, aat0995, jaa, kls4584,
     axt8828, maf8113, rxg8002, rew8422, yxl1461, rgh3595,
     axk0088, pal6640, 341-grd, sgu4018, mjl, pvf6609,
```

bmm3056, ash9053, jdb, wen2559, ajd4166, bdm, mcl4080,
cem9314, arl8258, ~cjm5578, hxp8813, hpb, ark, jbg7763,
wmc, mds1761, rxg1272, mjh1697, twc3796, kxt0498,
icss544, jst1734, txs3788, mpd5171, jfn, sxn, paw,
ajb0730, kcl8605, jah4007, ats, pxv3951, xxy4991, atk,
hxy8630, anhinga, cs3-grd, swm, kml4510, jeh, dmb3357,
jjm7570, ptt, gjn3855, wrc, waw, kmc6820, rlr6379, csx-
grd, adg1653, eh, csdoc, gxs5626, jjs9804, sxl2521,
afb, jck7765, emk3594, vxs5163, msw4585, mmr, jbo5112,
ajd0690, vm, ssg4819, jsb7384, rwd, bid5275, jkp3250,
mpv, abc8518, bxg6007, sjr1521, sph3228, jam1758,
mbp3753, tmh, sls6076, mjk8256, kxy6184, cslab