Extracting License Usage
Using Flexlm Logs

A Project Report

submitted by

KALYAN KUMAR V, EE14B065

in partial fulfilment of requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **Extracting License Usage Using Flexlm Logs**, submitted by **Kalyan kumar V, EE14B065**, to the Indian Institute of Technology Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Harishankar Ramachandran
Project Guide
Professor
Dept. of Electrical Engineering
IIT Madras, 600036

Place: Chennai

Date: 25 May 2018

ACKNOWLEDGEMENTS

This work would not have been possible without the guidance and the help of several people. I take this opportunity to extend my sincere gratitude to all those who made this thesis possible. First, I would like to thank all my teachers who bestowed me with good academic knowledge. I am indebted to my advisor Prof. Harishankar Ramachandran whose expertise, generous guidance and support made it possible for me to work on a topic that was of great interest to me. I would also like to thank Mrs. Gayathri P from CC Staff, IITM for helping me to understand the problem and arrive at a solution. I would like to thank my family for providing valuable support and motivation throughout my life. I would also like to thank all my friends and well-wishers for helping me during hard times and supporting me.

ABSTRACT

KEYWORDS: FlexNet, FLEXlm, licenses, Pay-per-use, Log files, Python

FlexNet Publisher(1) (formerly known as FLEXlm) is a software license manager from Flexera Software which implements license management. Currently FLEXlm is being used to implement license management at IIT Madras for various softwares such as Ansys, Abaqus, Matlab and Comsol. The licenses are allocated dynamically to machines, a license being checked-out when a user begins using the software on any given machine and checked-in when the user finishes using the software. In this way, we can support the user community of hundreds with limited number of licenses. Currently we can't track the usage of individual users and which departments they belong to. In order to implement pay-per-use policy for license usage, we need to find a way to find these statistics. This project showcases a Python based code design and implementation through which we extract the cost incurred to each user, each guide and each department, by using FLEXlm log files.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **ii**

LIST OF FIGURES **iv**

LIST OF CODE SEGMENTS **v**

1 Introduction **1**

2 Background **2**

3 Code Design **3**

4 Results **17**

4.1 Outputs 17

4.2 Plots 20

5 Conclusions **25**

5.1 Observations 25

5.2 Future Directions 25

5.3 Securities 26

LIST OF FIGURES

4.1 Department wise Ansys licenses issued 17
4.2 Department wise Ansys usage in seconds 17
4.3 Top 20 Ansys users 18
4.4 Top 10 Ansys users of mechanical dept 18
4.5 Professor wise Ansys usage in seconds 19
4.6 Department wise Ansys licenses issued 20
4.7 Department wise Ansys usage in minutes 20
4.8 Department wise Comsol licenses issued 21
4.9 Department wise Comsol usage in minutes 21
4.10 Department wise Abaqus licenses issued 22
4.11 Department wise Abaqus usage in minutes 22
4.12 Department wise Matlab licenses issued 23
4.13 Department wise Matlab usage in minutes 23
4.14 Department wise Matlab usage in minutes 24

Listings

3.1 Importing libraries and Opening files 3
3.2 Defining parameters 4
3.3 Searching for OUT's using RegEx 5
3.4 Updating user license usage and time stamps of OUT's 6
3.5 Searching for guide name using user-prof file 7
3.6 Updating profusage of corresponding guides 8
3.7 Searching for user names in institute id format 8
3.8 Searching for dept name using host name from dept-host file 9
3.9 Updating outs and depusage of corresponding departments 10
3.10 Searching for IN's using RegEx 11
3.11 Updating user license usage and time stamps of IN's 11
3.12 Searching for user names in institute id format 12
3.13 Searching for dept name using host name from dept-host file 13
3.14 Printing outs and license usage of all users 13
3.15 Printing license usage under each professor 14
3.16 Printing license usage of departments and top 20 users overall 14
3.17 Printing top 10 users of any department 15
3.18 Plotting the results 15

CHAPTER 1

Introduction

FLEXlm also known as FlexNet Publisher is a software license manager from Flexera Software which implements license management and is intended to be used in corporate environments to provide floating licenses to multiple end users of computer software. Computer software can be licensed in a variety of ways. A license to use a piece of software may be associated with a specific machine (node-locked), permitting it to only run on that machine (node in a network); alternatively, a company or an organization may buy a pool of floating licenses and these licenses may be allocated dynamically to machines, a license being checked-out when a user begins using the software on any given machine and checked-in when the user finishes using the software. In this way, for example, a company might buy a pool of 100 licenses but support a user community of hundreds of occasional users of the software (so long as no more than 100 users ever want to use the software simultaneously).

FLEXlm software(2) is a license management solution that allows software vendors to limit the number of software seats available to their users. FLEXlm supports different licensing policies such as Floating (aka Concurrent) and Node Locked licenses. These type of software systems are also called DRM Solutions (Digital Rights Management). Other software with the same functionality are SafeNet HASP, SafeNet RMS and IBM LUM. FLEXlm is used by a great variety of software vendors, such as ESRI, Autodesk, Mathworks, PTC, Ansys, Cadence and more. Here at IITM, we use floating or node-locked license policy for license management. We intend to implement pay-per-use policy for license usage but we don't have any means to track license usage statistically. We can extract usage data and statistics from log files using the code developed in this project.

CHAPTER 2

Background

The lmadmin daemon or License server manager receives the license request from the application and passes it to the vendor daemon. Vendor daemon is created by the admin and is unique to each FLEXlm enabled application installed on the network. It's job is to process the license request and grant/deny it based on number of licenses available and pre-defined license usage constraints. License file holds detailed license information such as the quantity and time extent of the license. The main focus of this project is on the Debug or Report logs. These are written by the license server manager(lmadmin), typically accounting license check-outs, check-ins and license denials.

```
12:18:15 (MLM) OUT: "MATLAB" ee16m084@ams112 (4 licenses)
```

Each line of the log files looks like the above line. The first part in hh:mm:ss format is the time stamp. The 'OUT' represents that the license is issued by the server to the user. 'IN' is displayed if the license is taken back from the user. 'DENIED' is displayed if the maximum number of licenses is reached. MATLAB is shown because the issued license is of MATLAB software. 'ee16m084' is the user name and 'ams112' is the host name that are registered on the license server. The number of licenses issued/taken is shown in the brackets and nothing is displayed if just one license is used.

We intend to design and implement a python based code that can parse through the log file and extract usage data of users and departments at IITM. As the log files have a standard format and doesn't differ much with applications, same code with some changes should work with logs of any other application/software registered on the license server. We also use an excel sheet that has department details of each host names, to get the department wise licenses issued and cumulative usage. We can find the number of licenses issued to each user and for each department, cumulative license usage of each user, each guide and for each department. With these statistics, one can formulate a pricing model based on pay-per-price policy at the interest of the institute.

CHAPTER 3

Code Design

The aim is to design a python based code to find license usage from FLEXlm logs. In this code, five things are computed parallelly: Total number of licenses issued to each user, Total number of licenses issued under each department, Cumulative license time usage of each user, Cumulative license time usage of each department and Cumulative license time usage under each guide. The structure and implementation of the code is explained below.

This part imports the required libraries and defines main function. matplotlib library is used to plot various graphs and datetime library is used to deal with time stamps of log files. In main function, lines, liness and linesss are three string arrays used to read lines of log file and department-host excel sheet and user-professor excel sheets respectively. These three files are passed as arguments to the main code. Error is printed in case of any IOError.

Listing 3.1: Importing libraries and Opening files

```
import sys
import re
import matplotlib.pyplot as plt     #import required libraries
import datetime as dt

def main():
    lines = []
    liness = []
    try:
        with open (sys.argv[1], "r") as file:
            lines=file.readlines()
            #to read each line of log file
        with open (sys.argv[2], "r") as filee:
            liness = filee.readlines()
            #to read each line of dept-host excel sheet
        with open (sys.argv[3], "r") as fileee:
            linesss = fileee.readlines()
            #to read each line of user-prof excel sheet
    except IOError as e:
        print e
        print "Usage: %s logfile" % sys.argv[0]
        #Print if IOError
        return 1
```

In this part, all the parameters required are defined. ’users’, ’features’ and ’hosts’ sets contain list of user names, softwares and host names respectively. ’outs’ dictionary refers to no. of licenses issued from each department. ’li’ dictionary refers to no. of licenses with each user. ’usage’, ’depusage’ and ’profusage’ dictionaries refer to cumulative license usage of each user, each department and each guide respectively.

Listing 3.2: Defining parameters

```
hosts = set()       #set of all hosts
users = set()       #set of all users
features = set()    #set of all features
outs = dict()       #dict of no. of licenses issued from each dept
li = dict()         #dict of no. of licenses with each user
time1 = dict()      #dict of current time stamps of each user
time2 = dict()      #dict of previous time stamps of each user
usage = dict()      #dict of cumulative license usage of each user
depusage = dict()   #dict of cumulative license usage of each dept
profusage = dict()  #dict of cumulative license usage under each professor
```

Here, we go through line by line in the log file and search for OUT’s, which basically means that license is issued to the user. ’lines’ is a string array which contains lines of log file and we pass each line as ’l’. The OUT lines will be in the format shown below.

```
hh:mm:ss (MLM) OUT: "feature" user@host (n licenses)
```

We use regular expressions grouping and matching to find OUT’s and store user name, host name, software name and time stamp of OUT in user, host, feature and time respectively. Then these are added to corresponding dictionaries and sets and are used in next parts of the code. We give department code (’me’ for mechanical department) as an argument to print top 10 license users from that department later. Also we can opt to print for any particular month alone by giving it as an argument (’Aug’ for August). Everyday, time stamps are printed in the format shown in RegEx below. When the required month starts, we start executing our code and end it when the log file reaches the next month. If we want to print for whole year, we give ’1’ as argument.

Listing 3.3: Searching for OUT’s using RegEx

```
department = str(sys.argv[4])
month = str(sys.argv[5])
p = 0
next = {'Jan':'Feb', 'Feb':'Mar', 'Mar':'Apr', 'Apr':'May', '↩
   May':'Jun', 'Jun':'Jul', 'Jul':'Aug', 'Aug':'Sep', 'Sep':'↩
   Oct', 'Oct':'Nov', 'Nov':'Dec', 'Dec':'Jan'}
if(month == "1"):
    month = r'\w+'
    next_month = "we are printing for whole year"
else:
    next_month = next[month]
for l in lines:
    m = re.search(r'\s' + month + r'\s[\d]{2}\s[\d]{4}\s', l, ↩
       re.I)
    if m:
        p = 1
    m = re.search(r'\s' + next_month + r'\s[\d]{2}\s[\d]{4}\s',↩
        l, re.I)
    if m:
        if(p == 1):
            p = 0
            break
    if(p == 1):
        # 12:18:15 (MLM) OUT: "MATLAB" ee16m084@ams112
        # 19:28:40 (MLM) IN: "MATLAB" shobhan@SRoy
        m = re.search(r'([^\s]+)\s+\((\w+)\)\s+OUT:\s+"(.+)"\s↩
           +([^\s]+@[^\s]+)', l)     #search for OUT's'
        if m:
            time = m.group(1)
            #get timestamp, feature, user and host using regex ↩
               grouping and add them to the dicts
            feature = m.group(3)
            user = m.group(4).split("@")[0]
            host = m.group(4).split("@")[1]
            hosts.add(host)
            features.add(feature)
            other = "other"
```

Using RegEx, we look for multiple licenses and store the count in 'lic'. If not found, initialize lic to one. Read the time stamp and convert it from string to a datetime element and store it in 'time2' dictionary with key as user name. The method here is that if previous time stamp is stored in 'time1', whenever there is an out, calculate the effective license usage between the two time stamps i.e the product of the number of licenses and the time difference(in seconds) and add it to the usage of corresponding user. Now, current time stamp of the user is stored in previous time stamp i.e time1. Now add lic to the li[user] i.e no. of licenses of user. If the user is new, then usage is initialized to zero and li[user] to lic.

Listing 3.4: Updating user license usage and time stamps of OUT's

```
        m = re.search(r'\((\d+)\s+licenses\)', l)
        #check if multiple licenses are issued
        if m:
            lic = int(m.group(1))
        else:
            lic = 1
            #if not just add 1 lecense

        time2[user] = dt.datetime.strptime(time, "%H:%M:%S") ↩
              #current time stamp. convertion from string to ↩
              datetime element

        if user in users:
        #if user is already present in users, add the effective license usage, also add the no. of licenses
            usa = (li[user])*((time2[user]-time1[user]).seconds)
            usage[user] += usa
            time1[user] = time2[user]
            li[user] += lic
        else:
            users.add(user)
            #else initiate usage to zero and update number of licenses
            usa = 0
            usage[user] = 0
            li[user] = lic
            time1[user] = time2[user]
            #current time stamp is stored in previous time stamp
```

We need to find the guide name from the user-prof file. But if we open the file in every loop, the code runs in O(n^2) time. So outside this loop, in the main function we parse through user-prof sheet and every line contains user names and guide names separated by spaces. We use RegEx grouping and searching to store these in a global dictionary named user-prof().

Listing 3.5: Searching for guide name using user-prof file

```
        for li in linesss:
    m = re.search(r'^([^\s]+)\s+([^\s]+)', li)
    if m:
        us = m.group(1)
        pr = m.group(2)
        user_prof[us.lower()] = pr
```

Now we have list of all user names mapped to corresponding guide names in the user-prof global dictionary. We use it now to update the profusage of that particular professor. As the user-guide details are not available now, we randomly assigned each user to one of 8 professors named ranging from ’A’ to ’H’. Once we get the real details of the file, we can pass it as an argument and get the original statistics. Similarly we do the same thing for IN’s also.

Listing 3.6: Updating profusage of corresponding guides

```
    if user.lower() in user_prof:
        if user_prof[user.lower()] not in profusage:
            profusage[user_prof[user.lower()]] = usa
        else:
            profusage[user_prof[user.lower()]] += usa
```

If the user name is institute id, then it matches with the format which is: first two letters represent department name, next two digits represent joining year, next letter represents degree and the final three digits represent the roll number. Our interest is in the first two letters that represent department name. We get it using RegEx searching and store it in ’dept’.Then we update outs[dept] and depusage[dept] with corresponding values from above.

Listing 3.7: Searching for user names in institute id format

```
            m = re.search(r'^([a-z]{2})[0-9]{2}[a-z][0-9]{3}$', ↩
               user)
            #search for user names in the institute id formats
            if m:
                dept = m.group(1)
                if not (dept in outs):
                #update outs and depusage of the corresponding dept
                    outs[dept] = lic
                    depusage[dept] = usa
                else:
                    outs[dept] += lic
                    depusage[dept] += usa
```

When the user name is not in institute id format, then we cannot get department name directly from user name. Then we use the dept-host excel sheet to get this info. This sheet contains department names for all host names. We could parse the sheet into ’lo’ and using RegEx, we can find the line which contains ’host’, thereby using RegEx again in that particular line to find out the department. But if we check every line of the sheet every time, the code runs in O(n^2) time.

So outside this loop, in the main function we parse through the dept-host sheet and every line contains department names and host names separated by spaces. We create global dictionary ’dept-host’ which stores department names for all host names. We use RegEx grouping and searching to do this. The dept-host sheet sometimes has host names along with ip addresses. So we remove ip using RegEx to get the required host name and corresponding department. Examples for aerospace and ocean are shown. Similarly we write for every department and give other as department name if it is unrecognizable.

Listing 3.8: Searching for dept name using host name from dept-host file

```
    for lo in liness:

        m = re.search(r'^(.+)\s([^\s]+)', lo)
        if m:
            A = m.group(1)
            B = m.group(2)
            m = re.search(r'^([^\s]+)@10', B)
            if m:
                B = B.split("@")[0]

            m = re.search(r'^([^\s]+)@192', B)
            if m:
                B = m.group(1)

            if re.search(r'aerospace', A, re.I):
                dept_host[B.lower()] = "ae"

            elif re.search(r'ocean', A, re.I):
                dept_host[B.lower()] = "oe"

            else:
                dept_host[C.lower()] = "other"
```

As we have list of all host names mapped to corresponding departments in dept-host global dictionary, we use it now to update the depusage and outs of the particular user. If it is not found in the dept-host sheet, then we just give ’other’ as the department name and update the values.

Listing 3.9: Updating outs and depusage of corresponding departments

```
            else:
                if(host in dept_host):
                    dept_name[user] = dept_host[host]
                    if not (dept_host[host] in outs):
                        outs[dept_host[host]] = lic
                        depusage[dept_host[host]] = usa
                    else:
                        outs[dept_host[host]] += lic
                        depusage[dept_host[host]] += usa

                else:
                    dept_host[host] = "other"
                    dept_name[user] = "other"
                    if not ("other" in outs):
                        outs["other"] = lic
                        depusage["other"] = usa
                    else:
                        outs["other"] += lic
                        depusage["other"] += usa
```

We use regular expressions grouping and matching to find IN's and store user name, host name and time stamp of IN in user, host and time respectively. Then these are added to corresponding dictionaries and sets and are used in next chunk of the code.

Listing 3.10: Searching for IN's using RegEx

```
        m = re.search(r'([^\s]+)\s+\((\w+)\)\s+IN:\s+"(.+)"\s+([^\s↩
            ]+@[^\s]+)', l)  #search for IN's
        if m:
            time = m.group(1)
            #feature = m.group(3)
            user = m.group(4).split("@")[0]
            host = m.group(4).split("@")[1]
            #hosts.add(host)
            #features.add(feature)
            other = "other"
```

Using RegEx, we look for multiple licenses and store the count in 'lic'. If not found, initialize lic to one. Read the time stamp and convert it from string to a datetime element and store it in 'time2' dictionary with key as user name. The method here is that if previous time stamp is stored in 'time1', whenever there is an out/in, calculate the effective license usage between the two time stamps i.e the product of the number of licenses and the time difference(in seconds) and add it to the usage of corresponding user. Now, current time stamp of the user is stored in previous time stamp i.e time1. Now subtract lic to the li[user] i.e no. of licenses of user as the licenses are taken back from user by the license server.

Listing 3.11: Updating user license usage and time stamps of IN's

```
    m = re.search(r'\((\d+)\s+licenses\)', l)
    #search for multiple licenses
            if m:
                lic = int(m.group(1))
            else:
                lic = 1

            time2[user] = dt.datetime.strptime(time, "%H:%M:%S") ↩
```

```
                        #current time stamp as datetime element

            if user in users:
            #if user already present in users, update usage and li
                usa = (li[user])*((time2[user]-time1[user]).seconds↩
                    )
                usage[user] += usa
                time1[user] = time2[user]
                li[user] -= lic
            else:
                print("error")
                usa = 0
```

We already showed how to search for user names in institute id format and we do the same here. Our interest is in the first two letters that represent department name. We get it using RegEx searching and store it in ’dept’.Then we update depusage[dept] with the value.

Listing 3.12: Searching for user names in institute id format

```
            m = re.search(r'^([a-z]{2})[0-9]{2}[a-z][0-9]{3}$', ↩
               user)
            #search for user names in institute id format
            if m:
                dept = m.group(1)
                if not (dept in depusage):
                    depusage[dept] = usa
                else:
                    depusage[dept] += usa
```

Similarly, if the user name is not in the institute id format, then the code checks the dept-host global dictionary and checks for the department to which the user belongs. Then it updates the depusage of that corresponding department. If not found, it assigns department as ’other’.

Listing 3.13: Searching for dept name using host name from dept-host file

```
            else:
                if(host in dept_host):
                    if not (dept_host[host] in depusage):
                        depusage[dept_host[host]] = usa
                    else:
                        depusage[dept_host[host]] += usa
                else:
                    if not (other in depusage):
                        depusage["other"] = usa
                    else:
                        depusage["other"] += usa
```

Now we have the total licenses issues to all departments stored in outs[] and cumulative license usage of all users in usage[]. We print them and now we have data of each and every user. The user can be charged based on cumulative license usage.

Listing 3.14: Printing outs and license usage of all users

```
    print("Department wise licenses issued:")
    for key in sorted(outs.iterkeys()):
    #print no. of dept wise licenses issued
        print ("%s: %s" % (key, outs[key]))

    print("\n")
    print("User wise usage in seconds:")
    for key in sorted(usage.iterkeys()):
    #print cumulative license usage of each user
        print ("%s: %s secs" % (key, usage[key]))

    print("\n")
```

We have total cumulative usage under each professor/guide stored in profusage[]. We print it to see the license usage statistics under each professor.

Listing 3.15: Printing license usage under each professor

```
    print("Professor wise %s usage in seconds:" % feature)
    for key in sorted(profusage.iterkeys()):
                    #print cumulative license usage of each prof
        print ("Prof %s: %s secs" % (key, profusage[key]))

    print("\n")
```

Similarly we have total cumulative usage of each department stored in depusage[]. We print it to see the department wise statistics and we can print top 20 users for the software on the license server by sorting usage[].

Listing 3.16: Printing license usage of departments and top 20 users overall

```
    print("Department wise usage in seconds:")
    for key in sorted(depusage.iterkeys()):
    #print cumulative license usage of each dept
        print ("%s: %s secs" % (key, depusage[key]))

    print("\n")
    p = 0
    print("Top 20 users:")           #print top 20 users
    for key, value in sorted(usage.iteritems(), key=lambda (k,v): (↩
       v,k), reverse = True):
        p += 1
        print("%s) %s: %s: %s secs" % (p, key, dept_name[key], ↩
           value))
        if(p == 20):
            break

    print("\n")
```

To find the top 10 users of any department,we follow this procedure. Lets see for mechanical department. We parse through the users set and check if the department name of that user is "me". Then we store the usage in different dictionary 'me-usage'. Then by sorting me-usage, we print the top 10 license users. Similarly we can do this for all departments, by giving the third input argument as the department code.

Listing 3.17: Printing top 10 users of any department

```python
print("Top 10 users of %s dept:" % department)
me_usage = dict()
for user in users:
    if(dept_name[user] == department):
        me_usage[user] = usage[user]
    else:
        continue
q = 0
for key, value in sorted(me_usage.iteritems(), key=lambda (k,v)↩
    : (v,k), reverse = True):
    q += 1
    print("%s) %s: %s secs" % (q, key, value))
    if(q == 10):
        break

print("\n")
```

Then we plot the number of licenses issued, license usage for each department and license usage under each guide using matplotlib library of python.

Listing 3.18: Plotting the results

```python
plt.bar(list(outs.keys()), outs.values(), color='b')
plt.title('Department wise %s licenses issued' % feature)
plt.xlabel('Department')
plt.ylabel('No. of %s licenses issued' % feature)
plt.show()

```

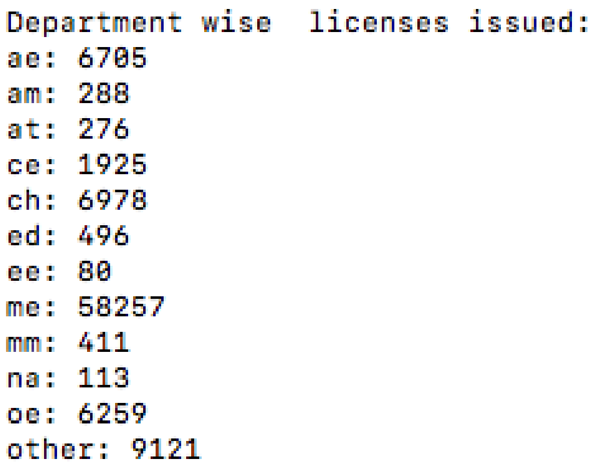
```
    plt.bar(list(depusage.keys()), depusage.values(), color='g')
    plt.title('Department wise %s usage in seconds' % feature)
    plt.xlabel('Department')
    plt.ylabel('%s Usage in seconds' % feature)
    plt.show()

    plt.bar(list(depusagemins.keys()), depusagemins.values(), color='b')
    plt.title('Department wise %s usage in minutes' % feature)
    plt.xlabel('Department')
    plt.ylabel('%s Usage in minutes' % feature)
    plt.show()

    plt.bar(list(profusage.keys()), profusage.values(), color='b')
    plt.title('Professor wise %s usage in seconds' % feature)
    plt.xlabel('Professor')
    plt.ylabel('%s Usage in seconds' % feature)
    plt.show()
```

CHAPTER 4

Results

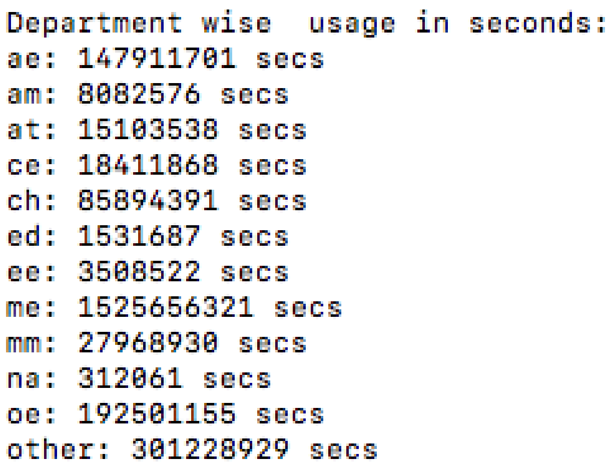
4.1 Outputs

The outputs printed, i.e department wise licenses issued and license usage in seconds for the Ansys log file is shown for the year 2017. So, we used '1' as the third argument to get statistics for the whole year. If we wanted statistics for the month of June, we give the third input argument as 'Jun'.



```
Department wise  licenses issued:
ae: 6705
am: 288
at: 276
ce: 1925
ch: 6978
ed: 496
ee: 80
me: 58257
mm: 411
na: 113
oe: 6259
other: 9121
```

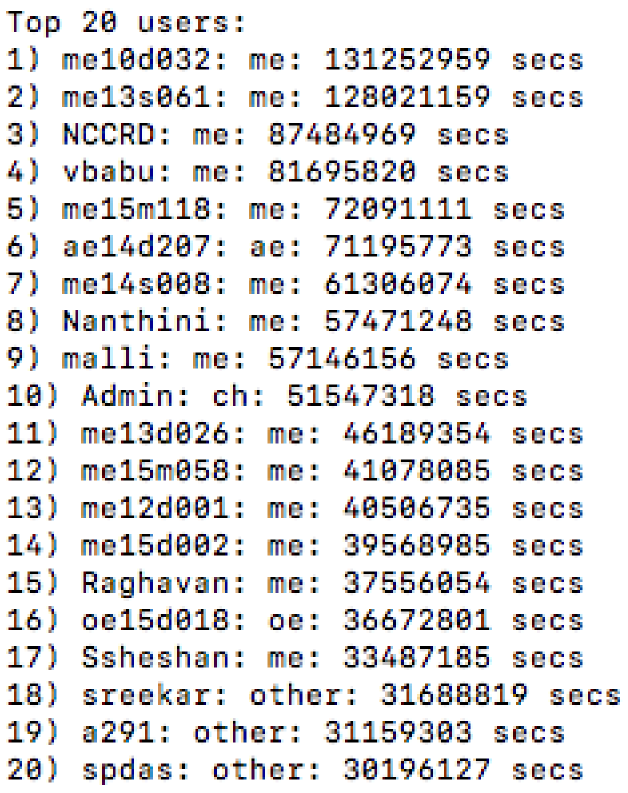
Figure 4.1: Department wise Ansys licenses issued



```
Department wise  usage in seconds:
ae: 147911701 secs
am: 8082576 secs
at: 15103538 secs
ce: 18411868 secs
ch: 85894391 secs
ed: 1531687 secs
ee: 3508522 secs
me: 1525656321 secs
mm: 27968930 secs
na: 312061 secs
oe: 192501155 secs
other: 301228929 secs
```

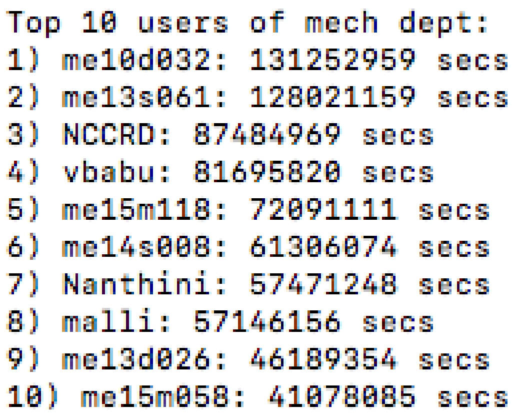
Figure 4.2: Department wise Ansys usage in seconds

Overall top 20 license server users of Ansys from all departments are shown below. Also Top 10 users from mechanical department are shown when the fourth argument is given as ’me’.



```
Top 20 users:
1) me10d032: me: 131252959 secs
2) me13s061: me: 128021159 secs
3) NCCRD: me: 87484969 secs
4) vbabu: me: 81695820 secs
5) me15m118: me: 72091111 secs
6) ae14d207: ae: 71195773 secs
7) me14s008: me: 61306074 secs
8) Nanthini: me: 57471248 secs
9) malli: me: 57146156 secs
10) Admin: ch: 51547318 secs
11) me13d026: me: 46189354 secs
12) me15m058: me: 41078085 secs
13) me12d001: me: 40506735 secs
14) me15d002: me: 39568985 secs
15) Raghavan: me: 37556054 secs
16) oe15d018: oe: 36672801 secs
17) Ssheshan: me: 33487185 secs
18) sreekar: other: 31688819 secs
19) a291: other: 31159303 secs
20) spdas: other: 30196127 secs
```

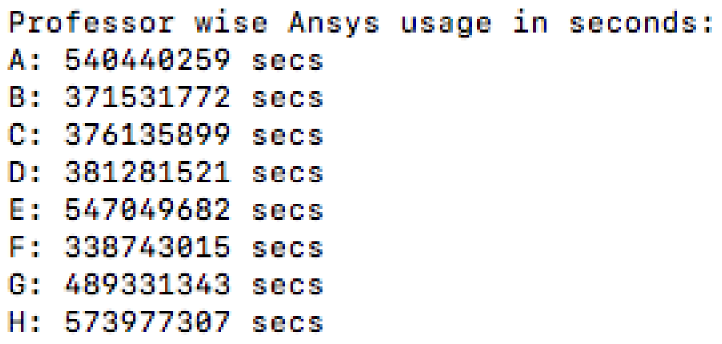
Figure 4.3: Top 20 Ansys users



```
Top 10 users of mech dept:
1) me10d032: 131252959 secs
2) me13s061: 128021159 secs
3) NCCRD: 87484969 secs
4) vbabu: 81695820 secs
5) me15m118: 72091111 secs
6) me14s008: 61306074 secs
7) Nanthini: 57471248 secs
8) malli: 57146156 secs
9) me13d026: 46189354 secs
10) me15m058: 41078085 secs
```

Figure 4.4: Top 10 Ansys users of mechanical dept

Currently we do not have the information of which user comes under which professor. For some softwares, we have the information but it is not logged in a format which we can use. Suppose some guide names are written in full names, some in short forms and some by their email id’s. hence we need to make sure that the professor details are logged in fixed format from here on. A good idea would be to use their institute email ids as they are unique. In this project we assigned each user to random professors, whose names range from ’A’ to ’H’. This is used for illustration purposes only.



```
Professor wise Ansys usage in seconds:
A: 540440259 secs
B: 371531772 secs
C: 376135899 secs
D: 381281521 secs
E: 547049682 secs
F: 338743015 secs
G: 489331343 secs
H: 573977307 secs
```

Figure 4.5: Professor wise Ansys usage in seconds

Also while running the script in terminal or command line, we use five arguments to run this script. First argument is the log file of the software for which we need the statistics. Second argument is the host-dept sheet which contains details of each host mapped to corresponding departments. Third argument is the user-guide sheet which contains details of each user mapped to corresponding guide. Fourth argument is the department code for which we need the top 10 license users from. Fifth argument is the three letter month for which we need the statistics for. If we need statistics for the whole year, we give the argument as ’1’.

4.2 Plots

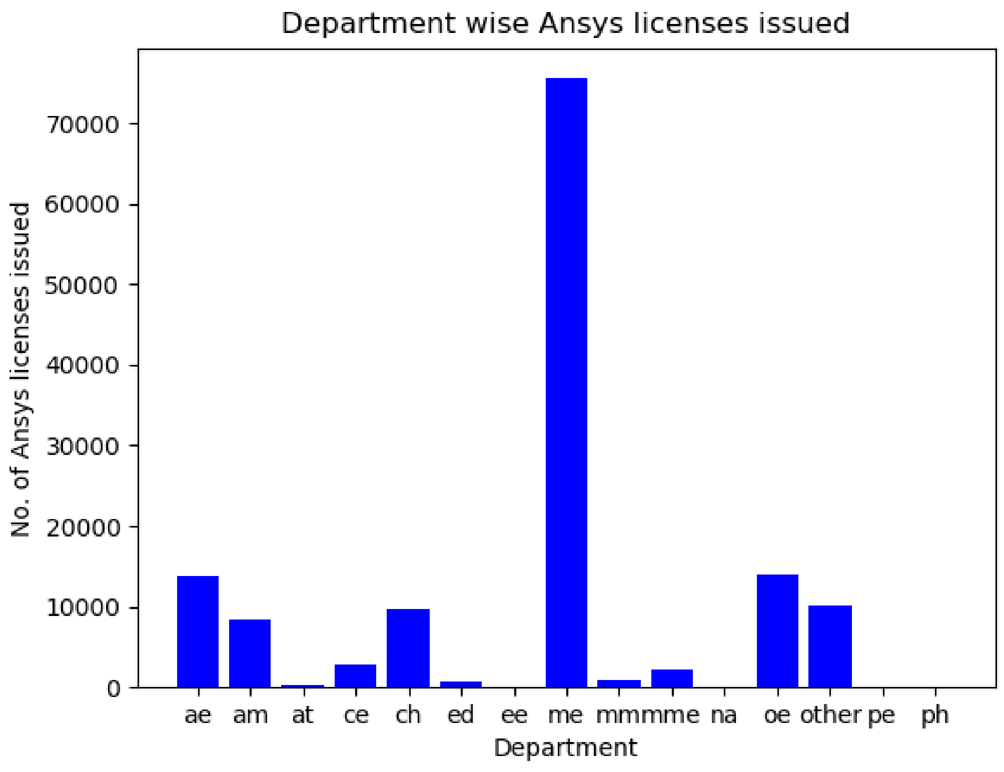


Figure 4.6: Department wise Ansys licenses issued

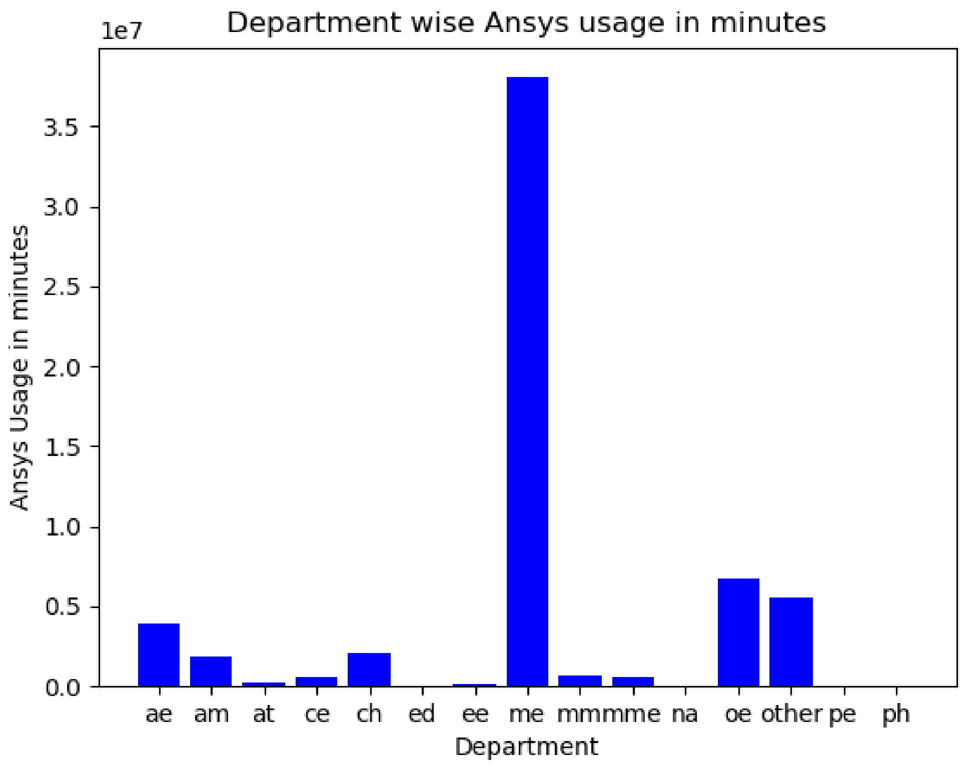


Figure 4.7: Department wise Ansys usage in minutes

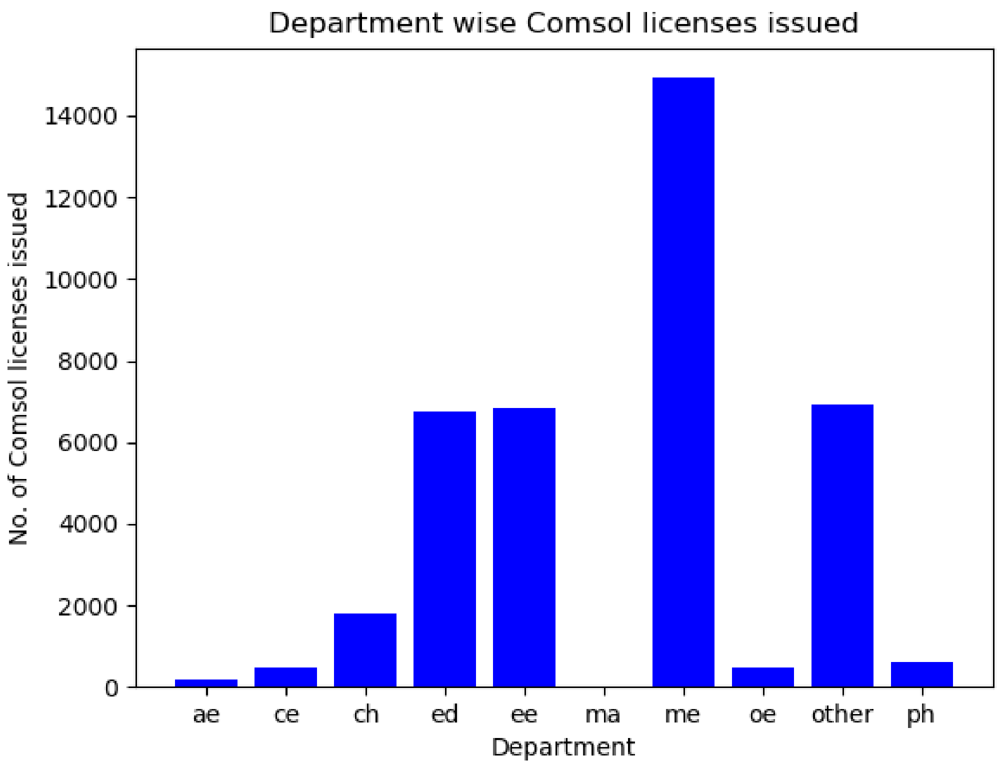


Figure 4.8: Department wise Comsol licenses issued

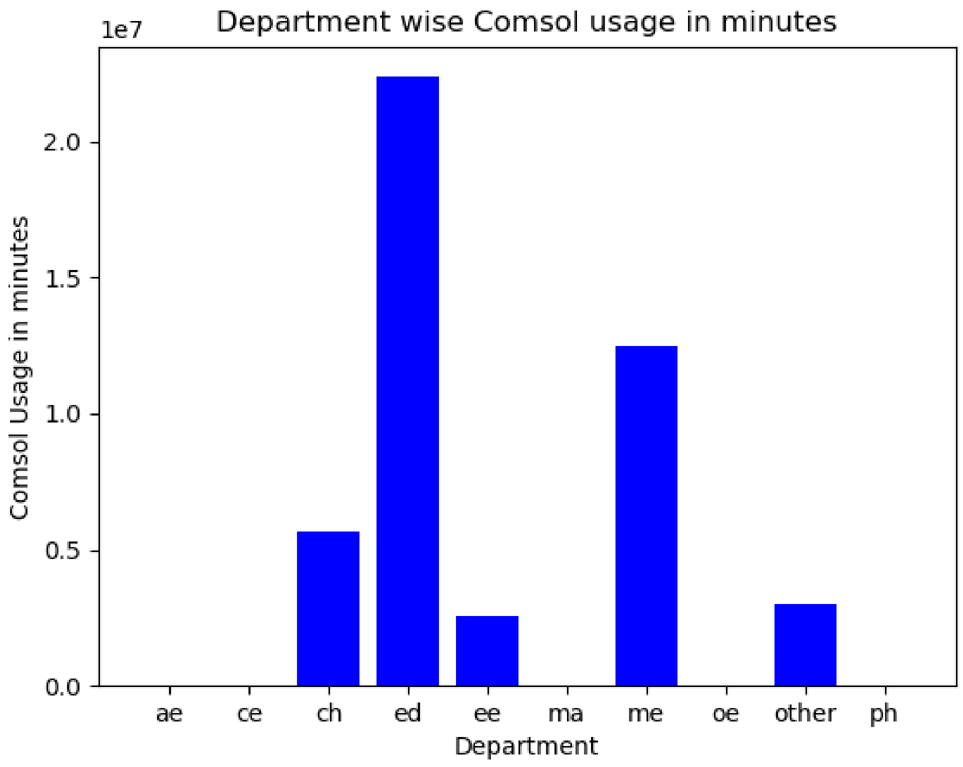


Figure 4.9: Department wise Comsol usage in minutes

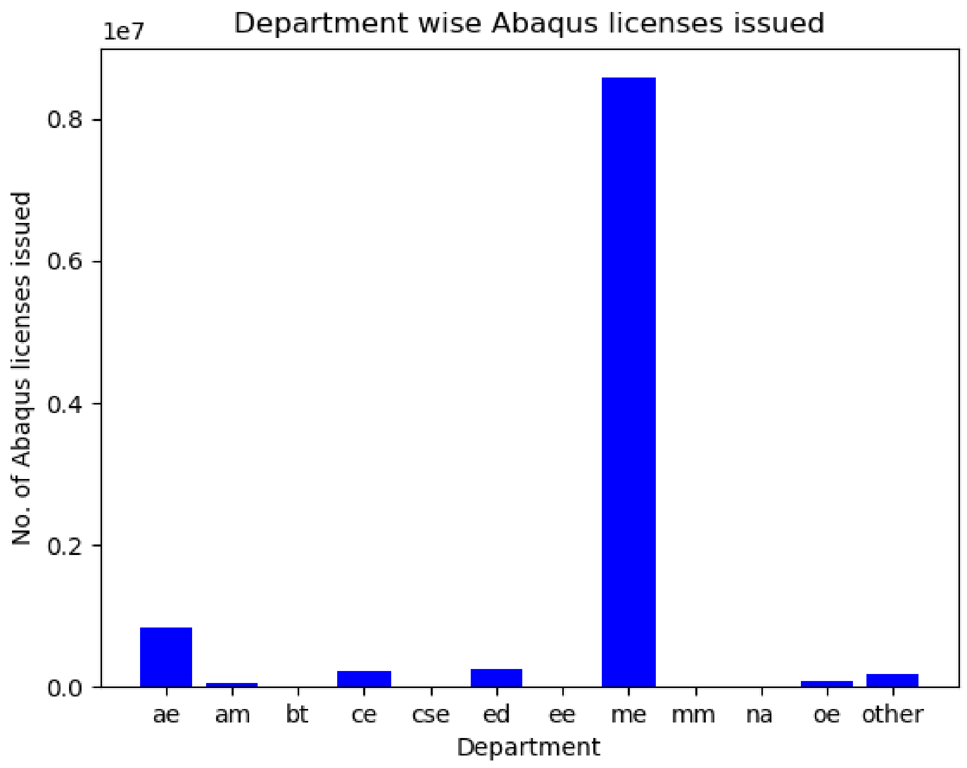


Figure 4.10: Department wise Abaqus licenses issued

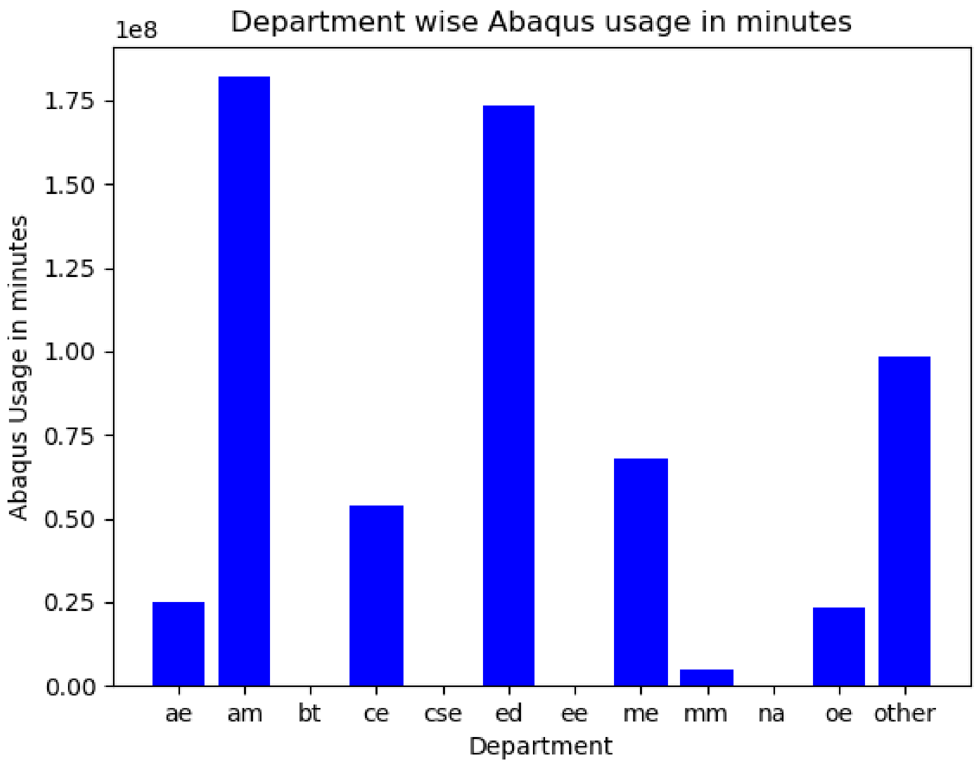


Figure 4.11: Department wise Abaqus usage in minutes

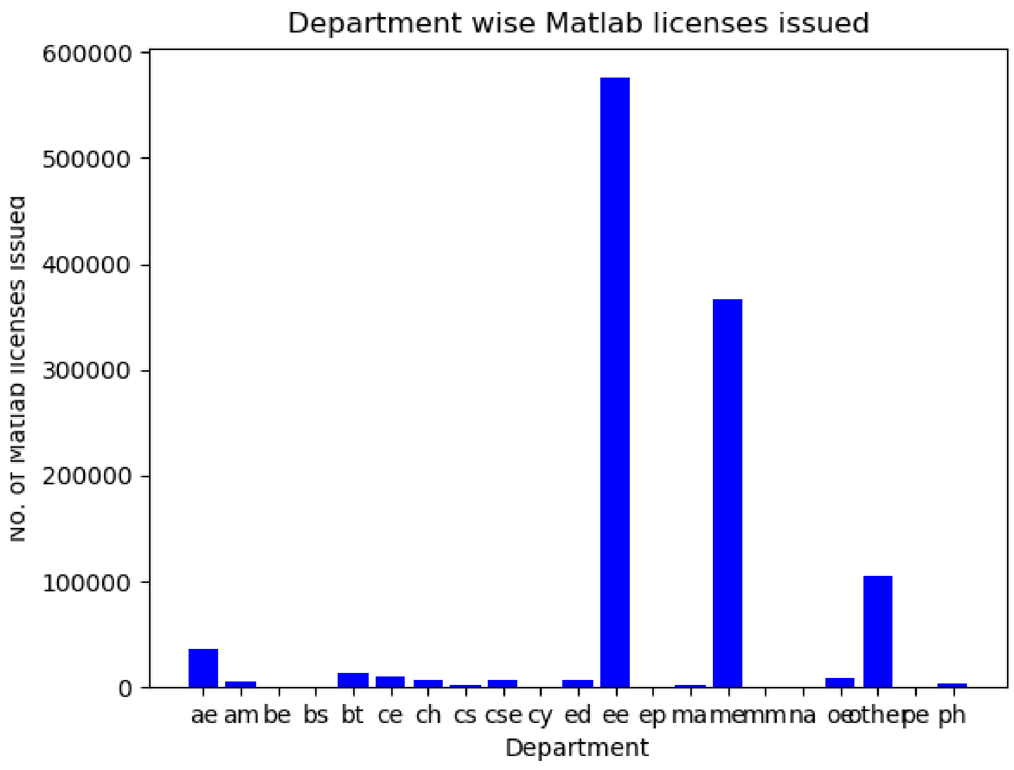


Figure 4.12: Department wise Matlab licenses issued

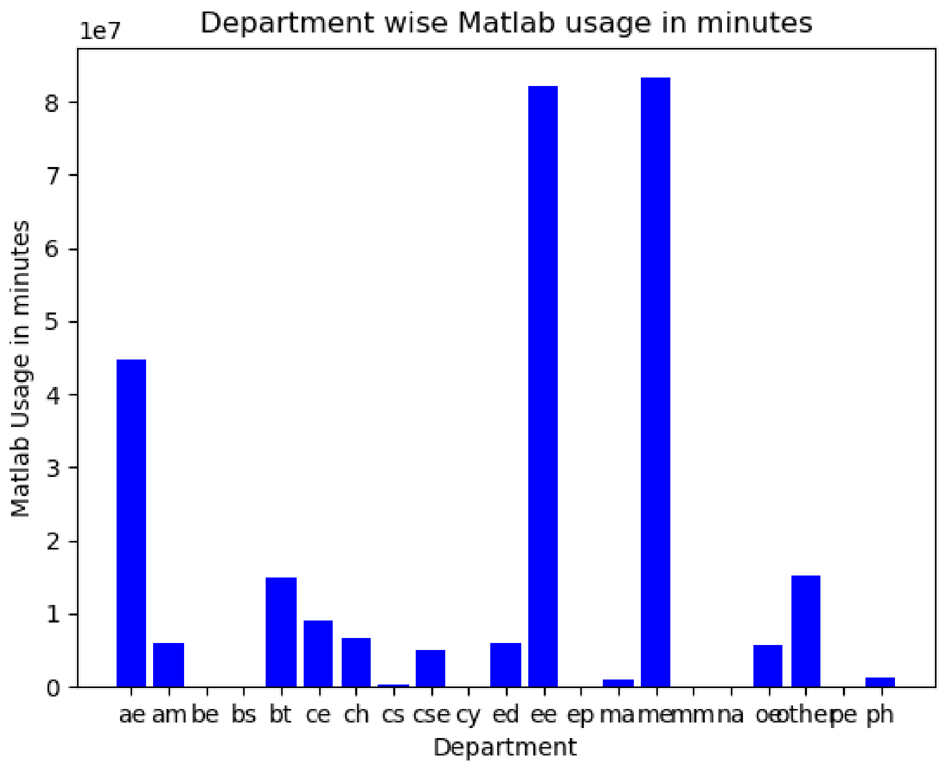


Figure 4.13: Department wise Matlab usage in minutes

After randomly assigning each user to a random professor, the license usage under each professor plot is shown below.

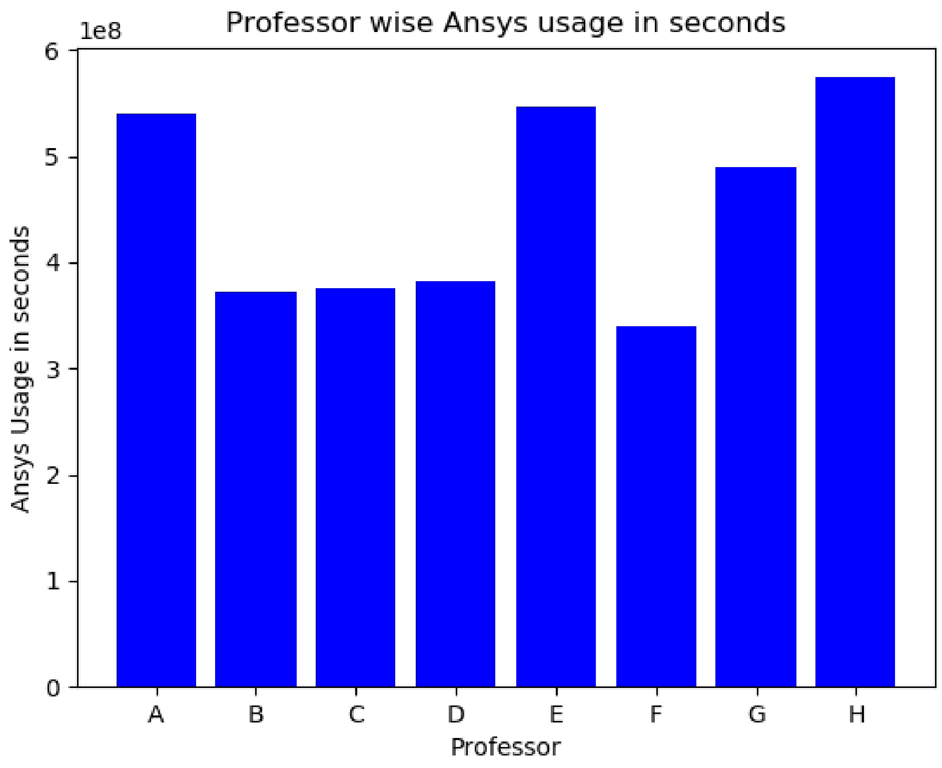


Figure 4.14: Department wise Matlab usage in minutes

CHAPTER 5

Conclusions

5.1 Observations

If we observe the outputs and plots, we can find some trends of users across various departments. For Ansys software, most of the users from Top 20 belongs to mechanical department. From the overall plots, we can see that the highest Ansys license usage comes from mechanical department. Similarly we can see the trends for other softwares. The users categorized under 'other' department has considerable license usage. This means that the user details are not updated in the host - department excel sheet. But by assigning those users under 'other' department, we store their usage as well. We get their details by printing the license usage of users using usage[] dictionary and selecting the specific users whose department is 'other', by using dept-name[] dictionary, which stores department names for each user. Then by checking the values printed here, we can find the details of the users from 'other' department and charge them based on their license usage.

5.2 Future Directions

Now we know the license usage based on user perspective and department perspective. Every user has a guide and they need the guide approval before registering on the license server. But currently the guide details are not being maintained properly. Some are left blank in the registrations excel sheet. Some users filled their guide name but we code cannot be written to sort based on guide names unless each guide name is given a unique id. Every professor working at IITM has a unique institute email id, this can be used as their guide's id while the users are registering on the license server. This way, we can get license usage under each guide as well. Similar to what we did to find the department usage, we can find the guide wise usage. To make the code 100 percent accurate, in future, we need to make sure that there are no typos in host names in the excel sheets.

5.3 Securities

It is our duty to make sure that license usage is not misused i.e someone should not be allowed to pretend to be someone else. While registering on the license server, the user is asked to fill his MAC address and IP address and the user is allowed to choose a unique user id or to use his default email id. Then the license is issued linked to that MAC and IP address only. Also while signing in to the software in browser, the software asks for your email id and password. So, it is the duty of the user to safeguard his passwords to prevent misuse of license usage.

REFERENCES

[1] FLEXlm - *Wikipedia contributors. (2018, April 13). FlexNet Publisher. In Wikipedia, The Free Encyclopedia. Retrieved May 24, 2018*

[2] FLEXlm operation - *What is FLEXlm? What is FlexNet? In OpenLM Blog. Retrieved May 24, 2018*