



**דו"ח סיכום לקורס מידול עומסים במערכות מחשב –
203.3713**

מוגש ל: אורנה אגמון בן-יהודה

מגישים:

- אליאס מרגילה ת.ז 318386646
- גוליאן שעאר ת.ז 316050608
- פהד נאסר ת.ז 318489168
- יובל תמייר ת.ז 201277894

תוכן עניינים

<u>2</u>	שלב 1 – בחירת TRACE
<u>3</u>	שלב 2
3	ניתוח ראשוני של המידע
3	NASA-Log File Graphs
9	MATLAB-Log File Graphs
11	מענה על שאלות
<u>15</u>	שלב 3 – התאמות בין התפלגיות לבין תוצאות ניתוח המידע
<u>16</u>	שלב 4 – התאמות בין גרפים לבין התפלגיות אמפיריות
16	NASA-Log File Graphs
16	CDF of Runtimes time – All users
18	CDF of Runtimes time – Other users
Error! Bookmark not defined.	CDF of Runtimes time – Special users
18	CDF of Interarrival time
20	MATLAB-Log File Graphs
20	CDF of Runtimes
22	CDF of Interarrival time
<u>24</u>	שלב 5 – התפלגות ZIPF
<u>27</u>	שלב 6
27	NASA-Log File
29	MATLAB-Log File
<u>30</u>	שלב 7 – יצירת HYPER DISTRIBUTION
<u>33</u>	שלב 8 – יצירת מודל המדמה את המידע הראשוני
<u>36</u>	שלב 9- אימות
<u>65</u>	ביבליוגרפיה ומקורות
<u>65</u>	COLOPHON

שלב 1 – בחרית Trace

בחרנו את ה-Trace של iPSC/860 NASA Ames. תוצאות ניתוח ה-log מוצגות בפירוט בהמשך הדוח.

בנוסף, מתואר בדוח גם הניתוח של ה-log הביתי של MATLAB.

שלב 2

ניתוח ראשוני של המידע

NASA-Log File Graphs

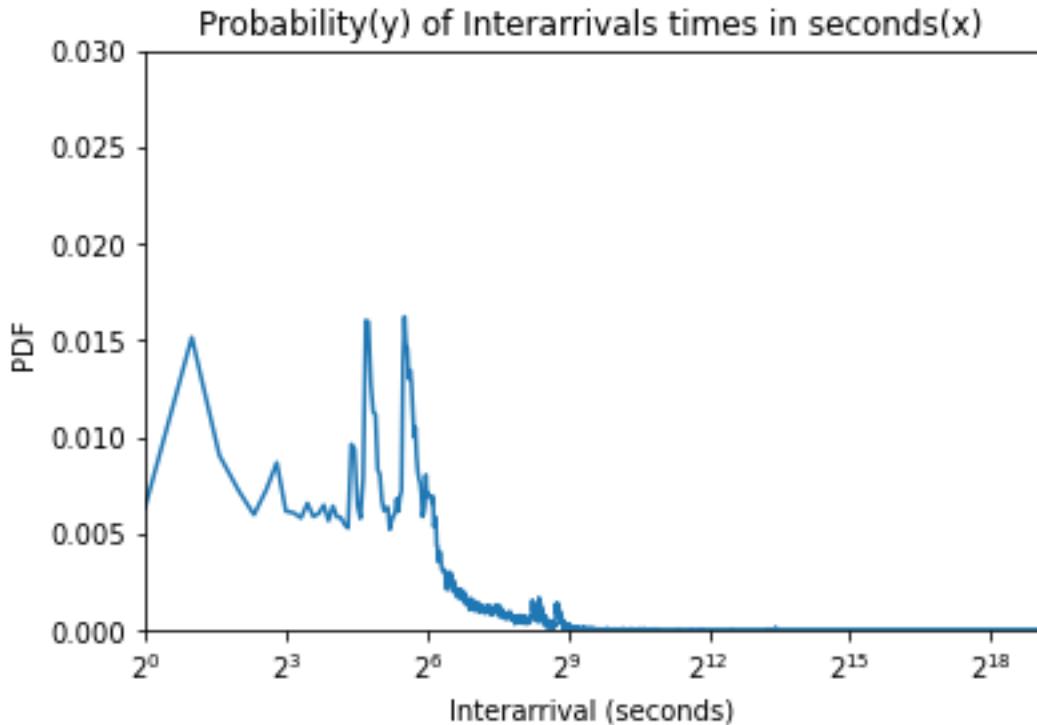


Figure 2-1

ערכי ה Interarrivals בציר האופקי

אנחנו יכולים לראות שהגרף מיוצג بصورة הלוגריתמית בציר ה- x והסיבה היא שהיו ערבים ממש גדולים שאנו יכולים לייצג עם הערבים הרגילים, בנוסף לכך הוא משתמש בעל שם משתמש "special" "shai" לו JOBS ארוכים במיוחד ומהם ממש כמעט ל-6 ימים. זה בעצם מסביר את ה *Heavy tail* שאפשר לראות בגרף.

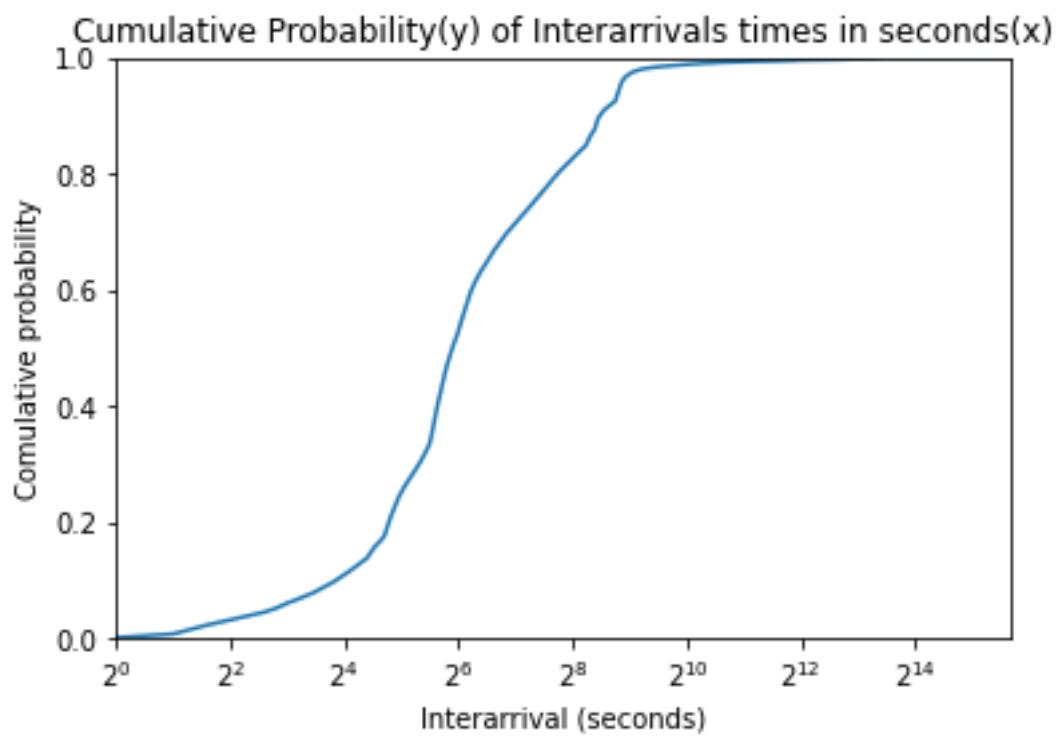


Figure 2-2

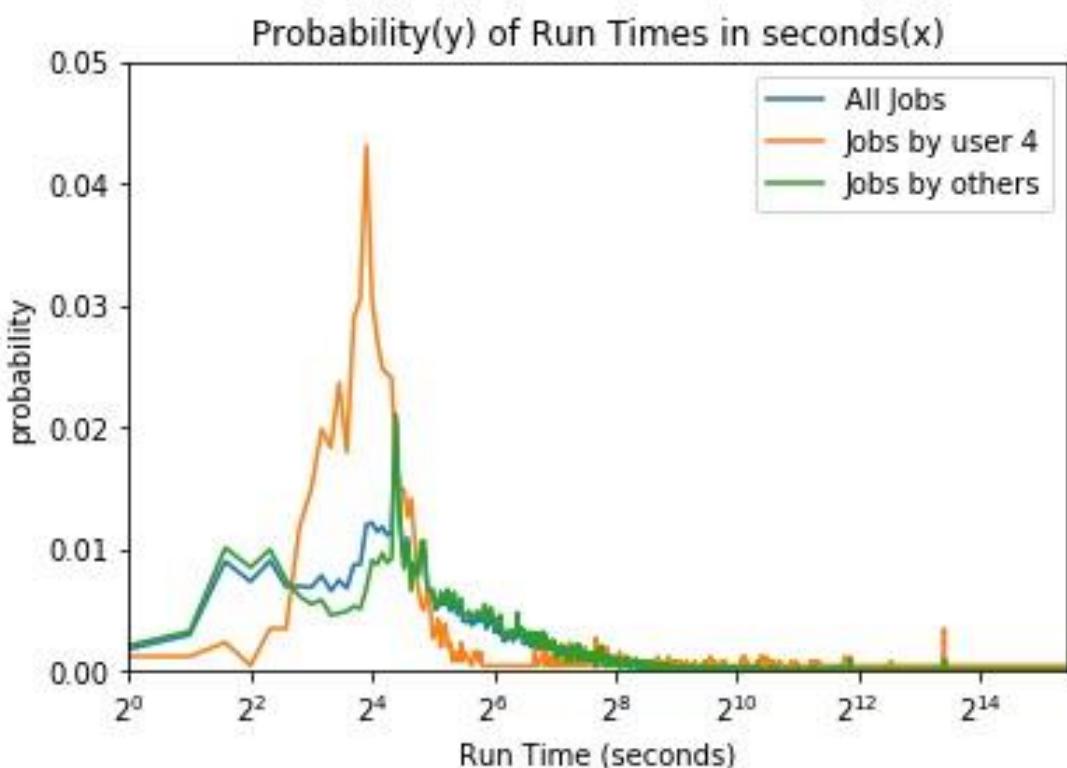


Figure 2-3

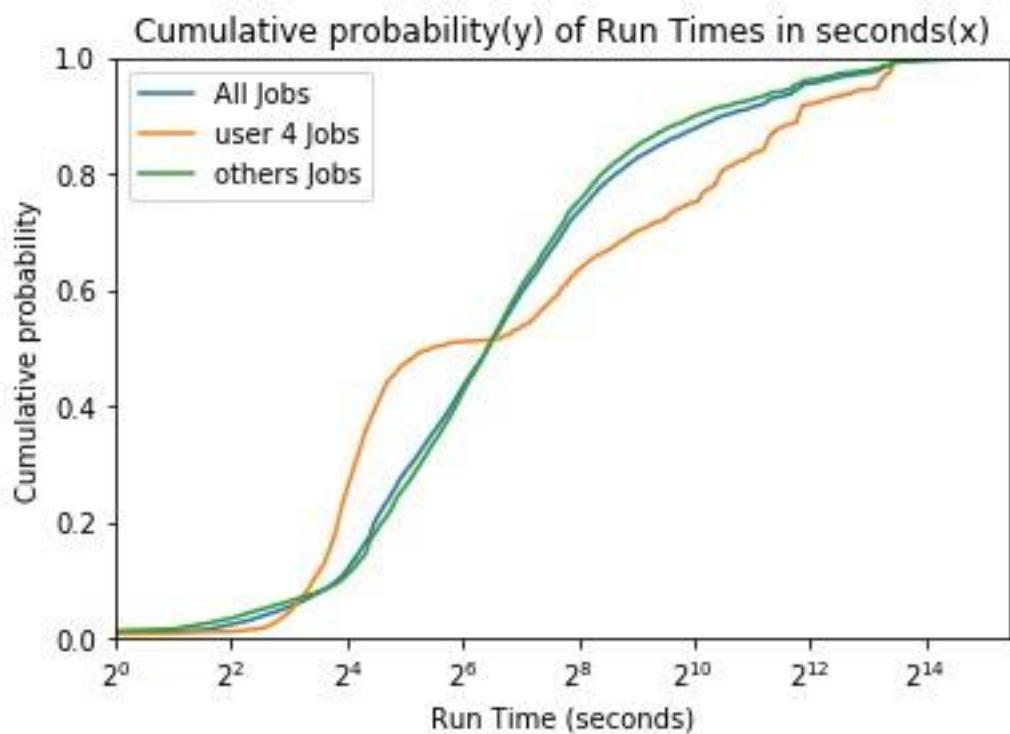


Figure 2-4

Run Time CDF גрафים שמראים התנהגות שונה של אחד הusers משאר הusers שזמן הריצה של הjobs שלו יותר גדולים משל השאר.

אפשר לראות גם שבזנב של הקו של משתמש "special" שאפילו כאשר $2^{14} > X$ את ה Y עדין לא שווה ל 1 וזה יצא מהתיבה שיש לא JOB שנמשך לפחות 6 ימים.

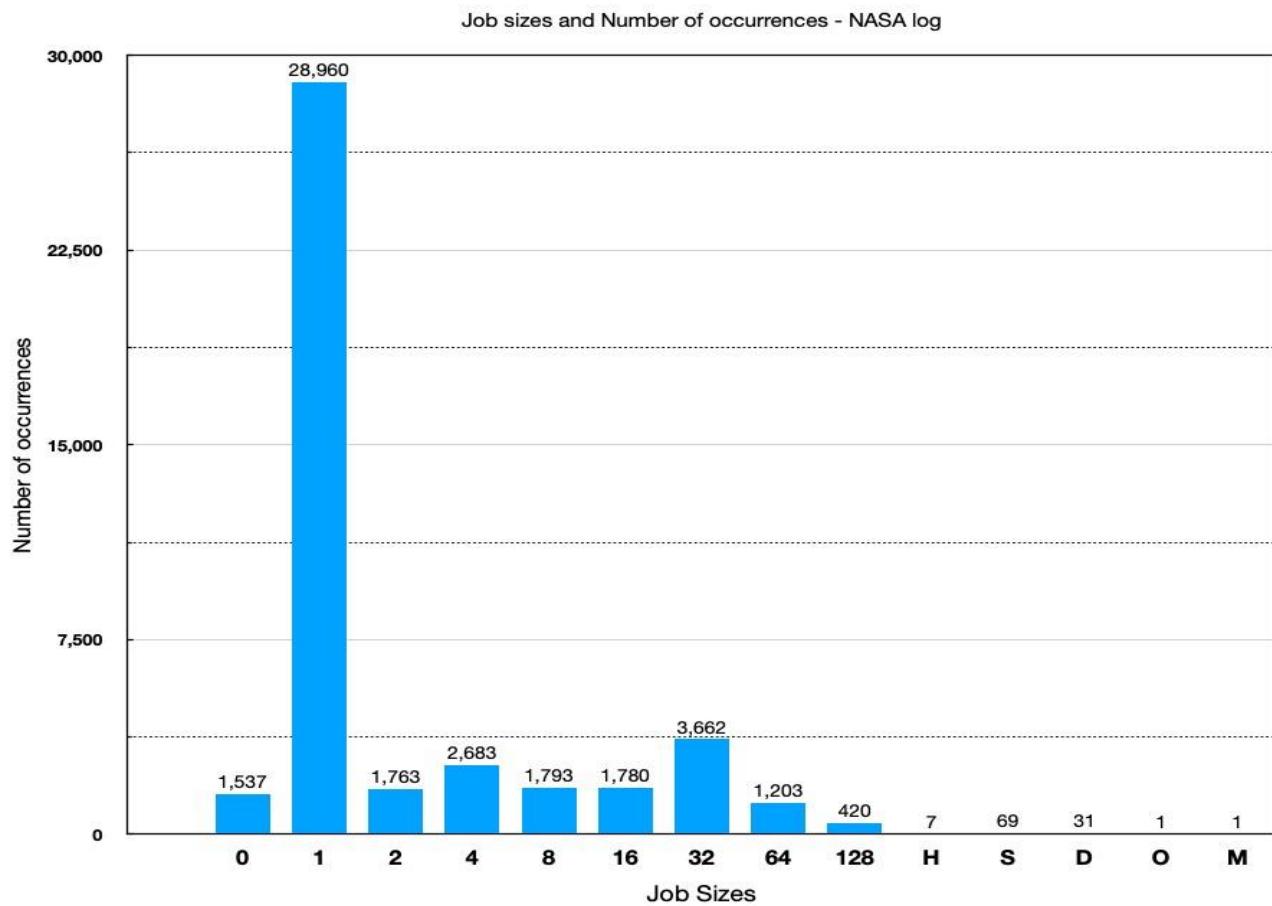
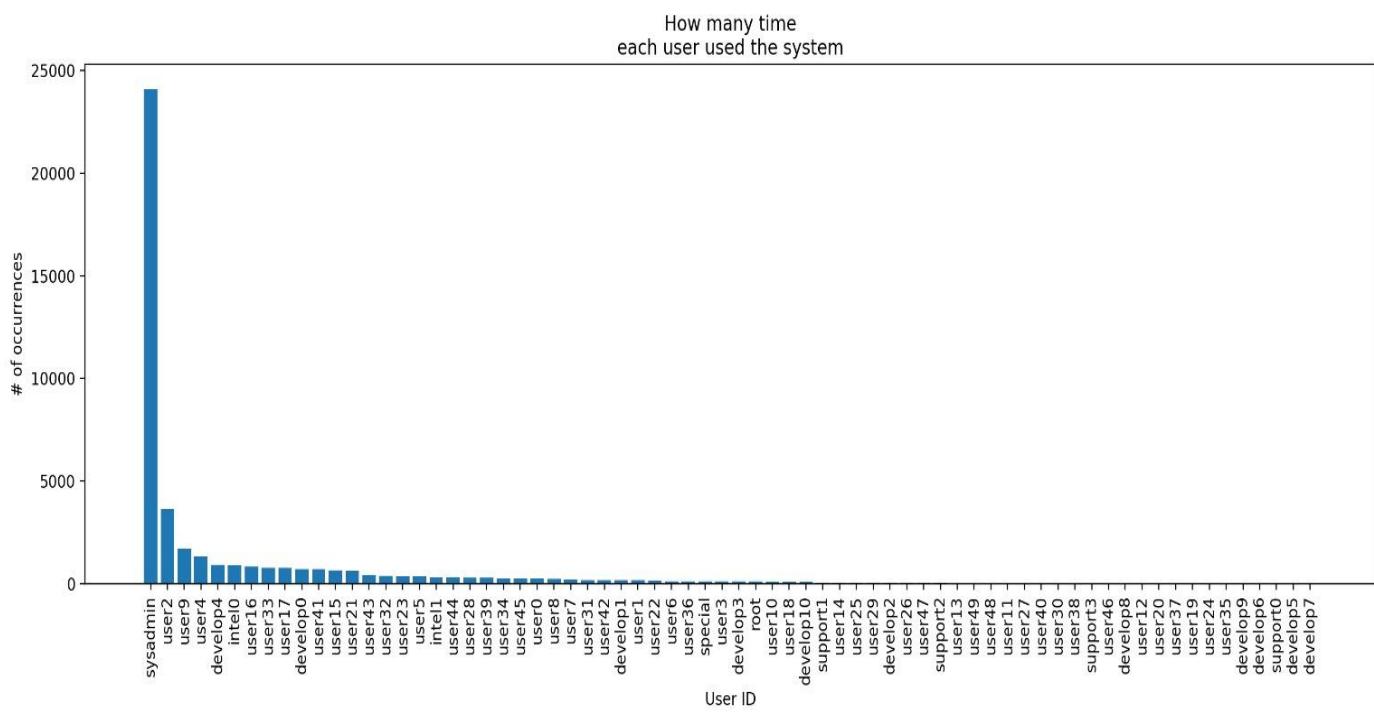
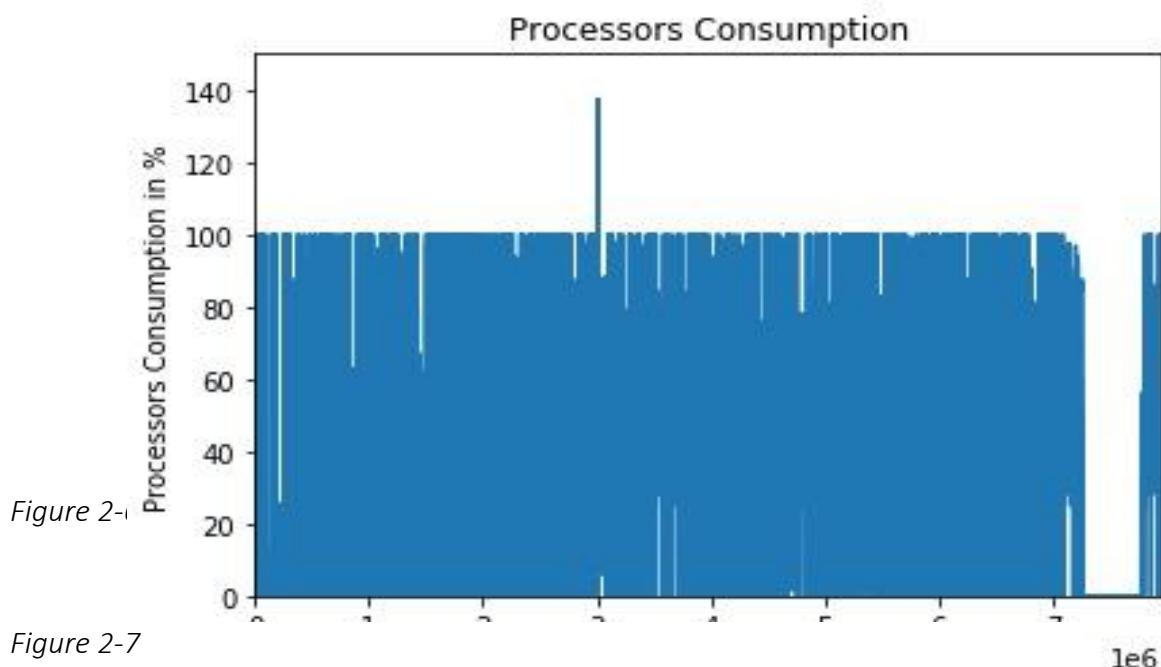


Figure 2-5

היסטוגרמה שומרה את מספר המופיעים של גודלי job שונים. אפשר לראות שהגודל הכי נפוץ הוא 1 וזה נובע מכך שיש המונ פועלות של sysadmin (אפשר לראות אותם בגרף הבא) שהם פועלות מערכת, אז בדרך כלל יהיו פועלות קצחות כמו למשל pid שבודקת פעילות המערכת.



האפק (Submit time) הגרף זהה מראה את אחוז המעבדים התפוסים בכל שנייה מההתחלה Log ועד סיום.



אפשר לראות את ה-Burst-Shmemca בגרף כאשר $\text{Submit time} \cong 3$, אחרי ניתוח عمוק יותר, הצלחנו לאתר את מקום התופעה. בדקנו מתי מספר המעבדים התפוסים עולה מעל 128 ואז הוציאנו את כל ה-Jobs שגרמו לתופעה לקרות. ה-Jobs שגרמו ל-Burst :

15855	3010043	-1	412	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15856	3010162	-1	24724	32	-1	-1	-1	-1	-1	-1	7	1	10	-1	-1	-1	-1
15857	3010217	-1	58929	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15858	3010264	-1	24626	32	-1	-1	-1	-1	-1	-1	7	1	10	-1	-1	-1	-1
15859	3010320	-1	58813	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15860	3010376	-1	25761	32	-1	-1	-1	-1	-1	-1	7	1	10	-1	-1	-1	-1
15861	3010441	-1	25698	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15862	3011133	-1	333	32	-1	-1	-1	-1	-1	-1	7	1	10	-1	-1	-1	-1
15863	3011191	-1	274	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15864	3011494	-1	333	32	-1	-1	-1	-1	-1	-1	7	1	10	-1	-1	-1	-1
15865	3011553	-1	273	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15866	3011837	-1	324	32	-1	-1	-1	-1	-1	-1	7	1	10	-1	-1	-1	-1
15867	3011892	-1	269	4	-1	-1	-1	-1	-1	-1	7	1	18	-1	-1	-1	-1
15868	3034897	-1	9357	64	-1	-1	-1	-1	-1	-1	4	1	-1	-1	-1	-1	-1

וזה אפשר לראות איך התופעה נוצרה, יותר מ job אחד שմבוך מעבדים הוגש תוך חלון זמן קצר יחסית עם זמני ריצה ארוכים מהרגיל. אך לא ידענו למי לחת את העדיפות להשתמש במעבדים.

(המספרים בציר ה-X הם כפولات של 10^6)

MATLAB-Log File Graphs

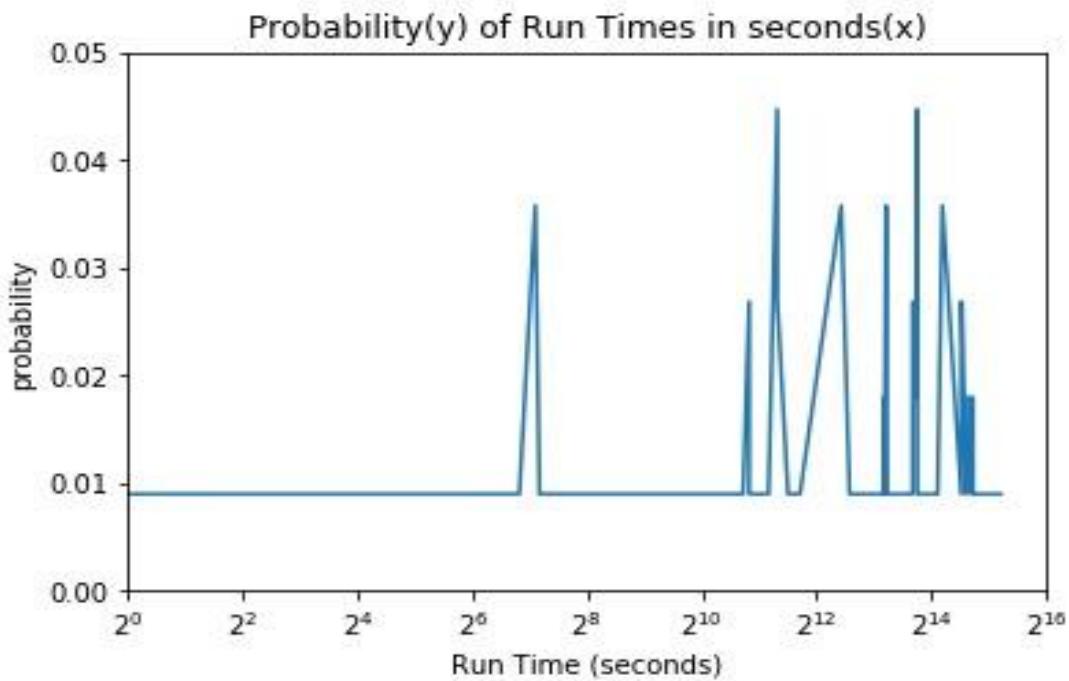


Figure 2-8

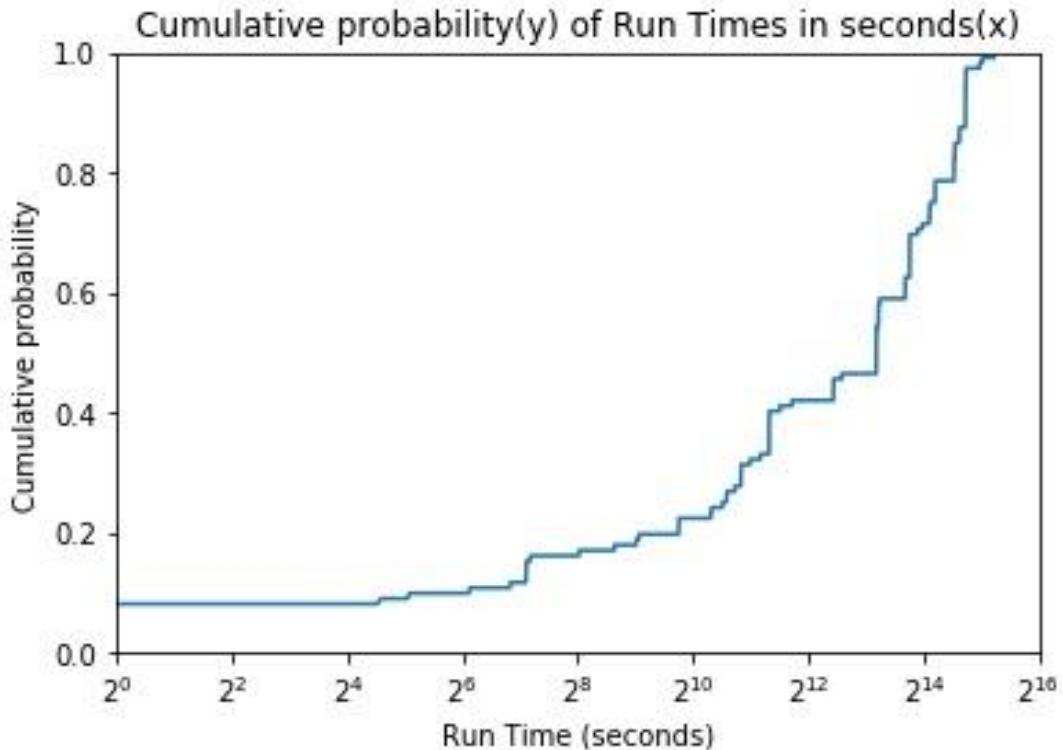
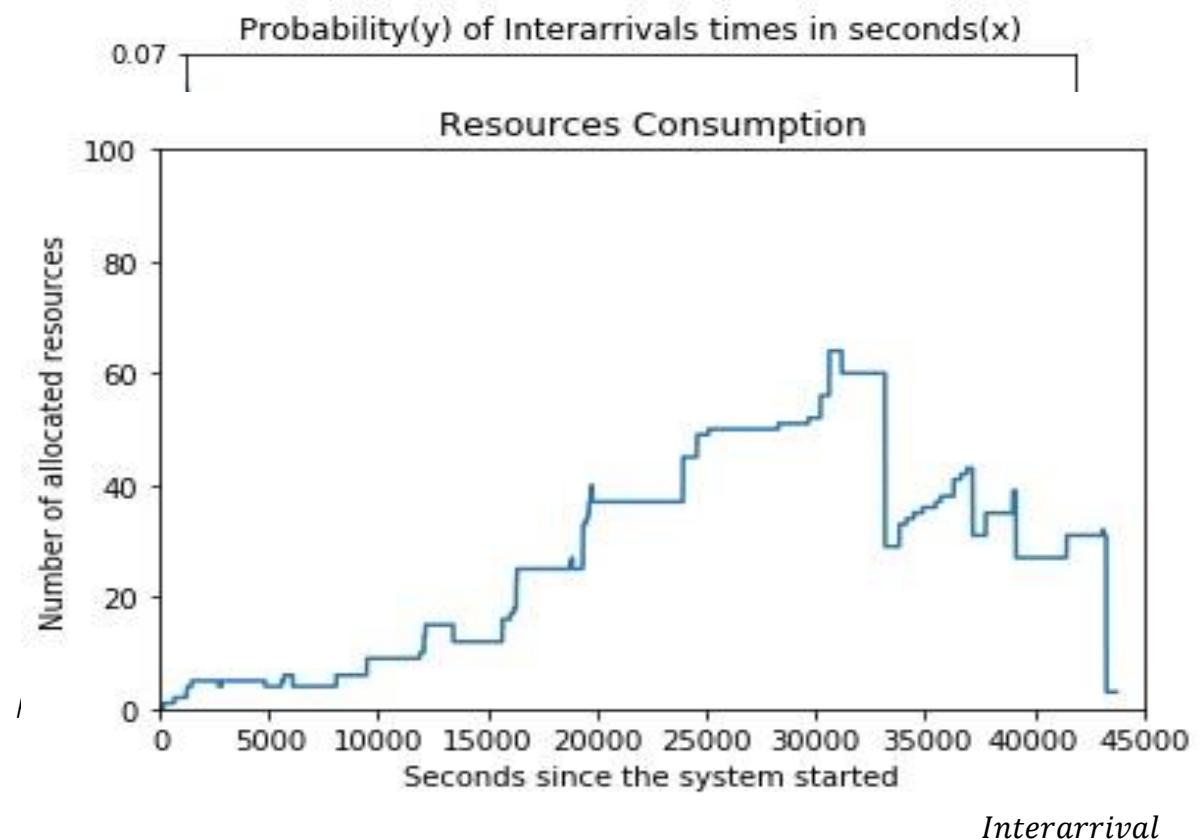


Figure 2-9

אפשר לראות שיש בכמה RunTimes נפוצות, חוץ מזה לא ניתן להסיק מסכימות מעכינות משום שה \log הוא קטן יחסית.



אפשר לראות בגרף ה PDF בכל מיני flurries ו bursts שבעצם מראות שיש כמה Interarrival Times נפרוצות.

אפשר גם לראות שכאשר $X=1$ יש לנו הסתברות גדולה יותר לשאר ה- α -values מאשר בעצם אומר לנו שיש כל מנה jobs שבאים אחד אחרי השני בשניה אחת.

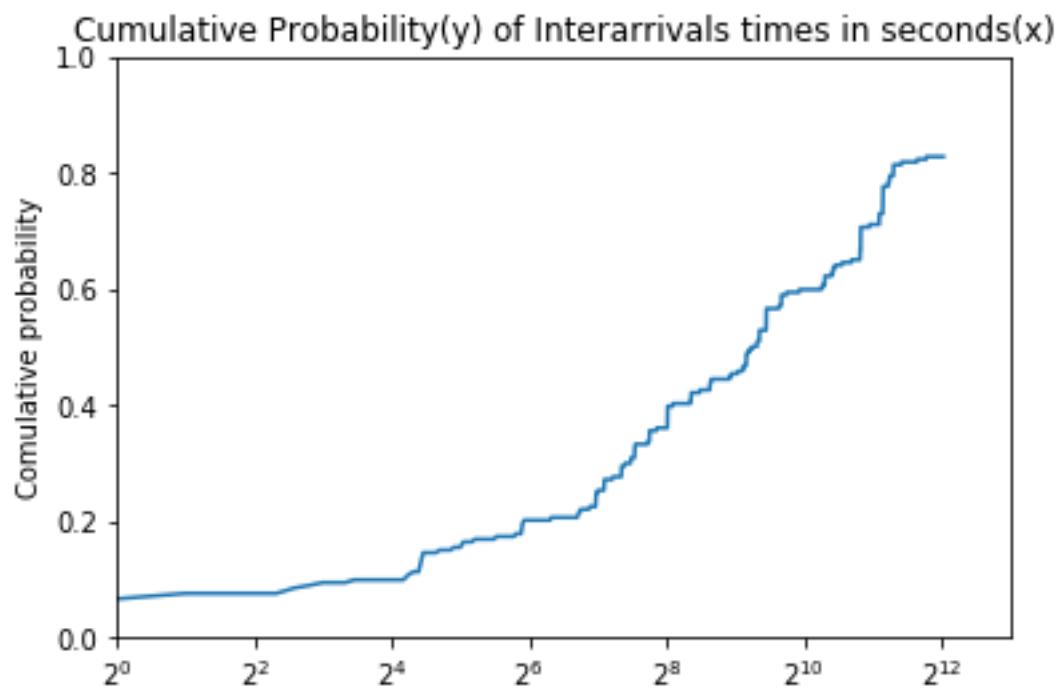


Figure 2-11

ניתן להסיק מהגרף ששיעור העומס על המערכת הינו בין 05:00-07:40 וזה הגיוני שהעומס ירד באופן משמעותי אחריו השעה 00:20 כי בדרך כלל גודל משתמשים (עובד היי-טק וסטודנטים) מסויימים עבדה בסביבות השעה זו.

או יותר נכון שהשתתת של MATLAB שחרר כל מיני משאביםividually ביחסו בזמן.

Figure 2-12

מענה על שאלות

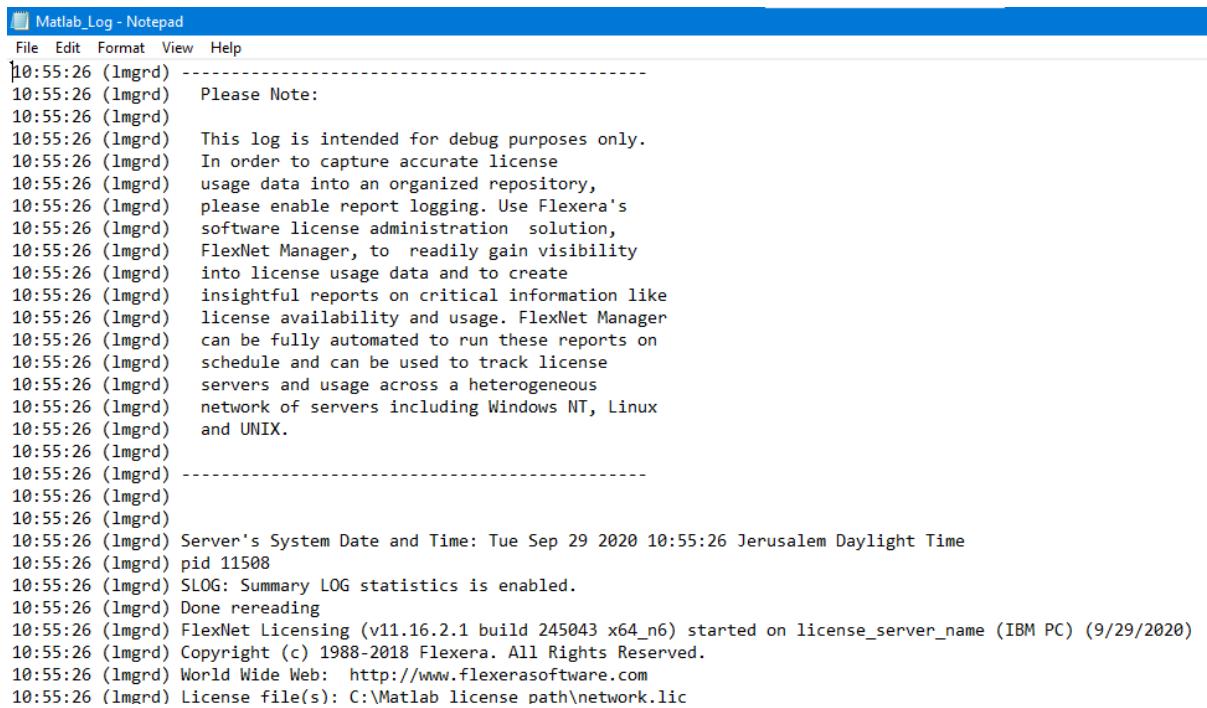
שאלה 2:

נקח למשל את הגרף של הלוג של NASA- Processors Consumption של הלוג של NASA- ניתן לראות כי יש לנו פיק של 140% של Processors שזאת תופעת Abnormal Behaviour. בגלל הסיבה הפוכה שבכל רגע נתון, הגרף הנ"ל מייצג את מספר המעבדים ה"תפוסים" ע"י הגובים השוכנים, לא ניתן ל"שים את האצבע" על הבעיה והג'וב הגרפי שגורם לפיק זהה. (בספר בעמוד מספר 50- פיסקה 2).

שאלה 3:

בטרייס של NASA, קיבלנו אותו נקי ומוכן לשימוש, אך לא הצטרכנו לעשות הרבה בנדון, אלא כבר ליישם את הנדרש (SWF, גרפים וכו').

בטריס של MATLAB שקיבלנו, אבן היה לנו כל מינו שורות שהיינו צריכים לנகות – בגן:



Matlab_Log - Notepad

File Edit Format View Help

```
10:55:26 (lmgrd) -----
10:55:26 (lmgrd) Please Note:
10:55:26 (lmgrd)
10:55:26 (lmgrd) This log is intended for debug purposes only.
10:55:26 (lmgrd) In order to capture accurate license
10:55:26 (lmgrd) usage data into an organized repository,
10:55:26 (lmgrd) please enable report logging. Use Flexera's
10:55:26 (lmgrd) software license administration solution,
10:55:26 (lmgrd) FlexNet Manager, to readily gain visibility
10:55:26 (lmgrd) into license usage data and to create
10:55:26 (lmgrd) insightful reports on critical information like
10:55:26 (lmgrd) license availability and usage. FlexNet Manager
10:55:26 (lmgrd) can be fully automated to run these reports on
10:55:26 (lmgrd) schedule and can be used to track license
10:55:26 (lmgrd) servers and usage across a heterogeneous
10:55:26 (lmgrd) network of servers including Windows NT, Linux
10:55:26 (lmgrd) and UNIX.
10:55:26 (lmgrd) -----
10:55:26 (lmgrd)
10:55:26 (lmgrd)
10:55:26 (lmgrd) Server's System Date and Time: Tue Sep 29 2020 10:55:26 Jerusalem Daylight Time
10:55:26 (lmgrd) pid 11508
10:55:26 (lmgrd) SLOG: Summary LOG statistics is enabled.
10:55:26 (lmgrd) Done rereading
10:55:26 (lmgrd) FlexNet Licensing (v11.16.2.1 build 245043 x64_n6) started on license_server_name (IBM PC) (9/29/2020)
10:55:26 (lmgrd) Copyright (c) 1988-2018 Flexera. All Rights Reserved.
10:55:26 (lmgrd) World Wide Web: http://www.flexerasoftware.com
10:55:26 (lmgrd) License file(s): C:\Matlab license path\network.lic
```

Figure 2-13

```
10:55:27 (MLM) Server started on license_server_name for: MATLAB
10:55:27 (MLM) SIMULINK MATLAB_5G_Toolbox AUTOSAR_Blockset
10:55:27 (MLM) Aerospace_Blockset Aerospace_Toolbox Antenna_Toolbox
10:55:27 (MLM) Audio_System_Toolbox Automated_Driving_Toolbox Bioinformatics_Toolbox
10:55:27 (MLM) Communication_Toolbox Video_and_Image_Blockset Control_Toolbox
10:55:27 (MLM) Curve_Fitting_Toolbox Signal_Blocks Data_Acq_Toolbox
10:55:27 (MLM) Database_Toolbox Datafeed_Toolbox Neural_Network_Toolbox
10:55:27 (MLM) Econometrics_Toolbox RTW_EMBEDDED_Coder Filter_Design_HDL_Coder
10:55:27 (MLM) Fin_Instruments_Toolbox Financial_Toolbox Fixed_Point_Toolbox
10:55:27 (MLM) Fuzzy_Toolbox GPU_Coder GADS_Toolbox
10:55:27 (MLM) Simulink_HDL_Coder EDA_Simulator_Link Image_Acquisition_Toolbox
10:55:27 (MLM) Image_Toolbox Instr_Control_Toolbox LTE_Toolbox
10:55:27 (MLM) MATLAB_Coder MATLAB_Builder_for_Java_Compiler
10:55:27 (MLM) MATLAB_Report_Gen MAP_Toolbox Mixed_Signal_Blockset
10:55:27 (MLM) MPC_Toolbox MBC_Toolbox Motor_Control_Blockset
10:55:27 (MLM) Navigation_Toolbox OPC_Toolbox Optimization_Toolbox
10:55:27 (MLM) Distrib_Computing_Toolbox PDE_Toolbox Phased_Array_System_Toolbox
10:55:27 (MLM) Powertrain_Blockset Pred_Maintenance_Toolbox RF_Blockset
10:55:27 (MLM) RF_Toolbox ROS_Toolbox Reinforcement_Learn_Toolbox
10:55:27 (MLM) Risk_Management_Toolbox Robotics_System_Toolbox Robust_Toolbox
10:55:27 (MLM) Sensor_Fusion_and_Tracking SerDes_Toolbox Signal_Toolbox
10:55:27 (MLM) SimBiology SimEvents SimDriveline
10:55:27 (MLM) Power_System_Blocks SimHydraulics SimMechanics
10:55:27 (MLM) Simscape_Virtual_Reality_Toolbox SL_Verification_Validation
10:55:27 (MLM) Simulink_Code_Inspector Real-Time_Workshop Simulink_Compiler
10:55:27 (MLM) Simulink_Control_Design Simulink_Coverage Simulink_Design_Optim
10:55:27 (MLM) Simulink_Design_Verifier Real-Time_Win_Target Simulink_PLC_Coder
10:55:27 (MLM) XPC_Target SIMULINK_Report_Gen Simulink_Requirements
10:55:27 (MLM) Simulink_Test SoC_Blockset Excel_Link
10:55:27 (MLM) Stateflow Statistics_Toolbox Symbolic_Toolbox
10:55:27 (MLM) System_Composer Identification_Toolbox Text_Analytics_Toolbox
10:55:27 (MLM) Trading_Toolbox Vehicle_Dynamics_Blockset Vehicle_Network_Toolbox
10:55:27 (MLM) Vision_HDL_Toolbox WLAN_System_Toolbox Wavelet_Toolbox
```

Figure 2-14

תיעוד של הרשת (התחלת) ושל הToolboxes הזמינים שנטענו.

```

14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:50:03 Jerusalem Daylight Time,#4,(3744K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:46:31 Jerusalem Daylight Time,#4,(3748K),(computer3,computer3,376)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:40:03 Jerusalem Daylight Time,#4,(3748K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:36:31 Jerusalem Daylight Time,#5,(3744K),(computer3,computer3,376)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:33:38 Jerusalem Daylight Time,#5,(3740K),(user7,computer7,696)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:30:03 Jerusalem Daylight Time,#5,(3740K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:26:31 Jerusalem Daylight Time,#5,(3744K),(computer3,computer3,376)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:23:38 Jerusalem Daylight Time,#5,(3744K),(user7,computer7,696)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:20:03 Jerusalem Daylight Time,#5,(3744K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000000 ms at Tue Sep 29 2020 14:18:52 Jerusalem Daylight Time,#5,(3688K),(user7,computer7,696)
14:55:28 (MLM) (@MLM-SLOG@)
14:55:28 (MLM) (@MLM-SLOG@) === Top 10 Peak Client Requests Processing Time (in ms) ===
14:55:28 (MLM) (@MLM-SLOG@) Time: Tue Sep 29 2020 14:55:28 Jerusalem Daylight Time
14:55:28 (MLM) (@MLM-SLOG@) Request processing time, when, #concurrent clients, (private bytes (in KB)), client info (user, node, FD)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 10:58:46 Jerusalem Daylight Time,#3,(3976K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 10:55:29 Jerusalem Daylight Time,#3,(3900K),(user8,license_server_name,640)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 10:58:45 Jerusalem Daylight Time,#3,(3976K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 10:58:45 Jerusalem Daylight Time,#3,(3976K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 10:58:48 Jerusalem Daylight Time,#3,(3976K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 10:58:51 Jerusalem Daylight Time,#3,(3976K),(user1,computer1,644)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 11:17:44 Jerusalem Daylight Time,#5,(3744K),(computer3,computer3,376)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 11:18:18 Jerusalem Daylight Time,#5,(3688K),(computer3,computer3,376)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 11:20:27 Jerusalem Daylight Time,#6,(3688K),(user4,computer4,696)
14:55:28 (MLM) (@MLM-SLOG@) 00000016 ms at Tue Sep 29 2020 11:20:32 Jerusalem Daylight Time,#6,(3688K),(user4,computer4,696)
14:55:28 (MLM) (@MLM-SLOG@)
14:55:28 (MLM) (@MLM-SLOG@) === Top 10 Peak In-house Operations time (in ms) ===
14:55:28 (MLM) (@MLM-SLOG@) NOTE: Peak times greater than 4 seconds get recorded.
14:55:28 (MLM) (@MLM-SLOG@) Time: Tue Sep 29 2020 14:55:28 Jerusalem Daylight Time
14:55:28 (MLM) (@MLM-SLOG@) In-house operation time, when, #concurrent clients
14:55:28 (MLM) (@MLM-SLOG@)
14:55:28 (MLM) (@MLM-SLOG@) === Active Connections Info ===
14:55:28 (MLM) (@MLM-SLOG@) Peak active connections #6 attempted at Tue Sep 29 2020 11:20:01 Jerusalem Daylight Time
14:55:28 (MLM) (@MLM-SLOG@)
14:55:28 (MLM) (@MLM-SLOG@) === Memory Usage Info ===
14:55:28 (MLM) (@MLM-SLOG@) Peak private bytes 3984K attempted at Tue Sep 29 2020 10:58:33 Jerusalem Daylight Time

```

Figure 2-15

סיכום מעט לעת של המשתמשים ושל החיבורים הפעילים במערכת ברגע.

כמובן שלאחר הניקוי, היה לנו מידע נטו שנוכל להשתמש בו, לצורך בניית ה-SWF, כי כל שורה שהיא מהצורה של IN/OUT השארנו, כך שנוכל לקבל את מלאה המידע הדורש במשימה הזאת ולמשימות הבאות.

אין בכלל מה להשוות בין התוצאות שהיינו מקבלים בlij הניקוי לעיל, כי כל מה שניקינו – יכולם לקבל מהמידע שנשאר (חיבורים פעילים, שמות משתמשים, שעת התחלת סיום וכו'...).

לכן, התוצאה הסופית של הלוג הנקי הייתה רק שורות של IN OUT כך שנוכל לדעת עברו כל משתמש וכל משאב מי, מתי וכמה השתמשו בו למשל.

שלב 3 – התאמות בין התפלגות לבון תוצאות בתחום המידע

נתחיל להתאים כל גרפף להתפלגות מסוימת.

Figure 2-1 - Multiple Erlang distributions:

ניתן לראות שגרף זה "מורכב" משתי התפלגות Erlang, כאשר הפרמטר k יהיה $2 = .$

Figure 2-2 - CDF of Erlang distribution:

באוטו אופן לגרף מס' 1, נראה כי גרפף זה מייצג לנו CDF של התפלגות Erlang כאשר הפרמטר k יהיה $= 2.$

Figure 2-3 - with tail of exponential distribution: More than one Hyper-Erlang

ניתן לראות שגרף זה הינו שילוב של שתי התפלגות – התפלגות Erlang של user רגיל והתפלגות Erlang של user special. לכן, שילוב של שתיהן ייתן לנו Hyper-Erlang distribution, כאשר הפרמטרים יהיו $k = 1, \quad p = P(\text{user}), \quad 1 - p = P(\text{special user})$

Figure 2-4 - CDF of Hyper-Erlang:

באוטו אופן לגרף מס' 3, נראה כי גרפף זה מייצג לנו CDF של התפלגות Hyper-Erlang כאשר ערכי הפרמטרים זהים לgraf ה-PDF.

Figure 2-5 אין התפלגות שמתאימה להיסטוגרמות

Figure 2-6 אין התפלגות שמתאימה להיסטוגרמות

Figure 2-7 אין התפלגות שמתאימה להיסטוגרמות

Figure 2-8 Mixed lognormal:

ניתן לראות שגרף זה מתפלג בצורה "יחסית" נורמלית, אך גרפף זה נמצא במרחב לוגרייטמי (ציר ה- x הינו סקללה לוגריתמית), ולכן גרפף זה מתאים להתפלגות לוג-נורמלית. בנוסף נשים לב שה-mean וה-standard deviation נמצאים במרחב לוגרייטמי.

Figure 2-9 -

לפי המידע על התפלגות הלוג-נורמלית, אין צורה חד-משמעית לגבי ה-CDF.

Figure 2-10 -

גרף זה מטעה קלות – ראשית, נשים לב שהוא נמצא במרחב לוגרייטמי. אם נסתכל על ה-interval שנמצא מ- $2^{12} - 2^4$, ניתן לחשב שמדובר בהתפלגות לוג-נורמלית (מרחב לוגרייטמי, התפלגות "יחסית" נורמלית). מצד שני, ה-interval שנמצא בתחלת הגרפף עד 2^4 לא ברור לగמרי.

Figure 2-12 - Normal distribution:

ניתן לראות בgraf זה "עקומת פעמון" אשר מאפיינת את העומס על המערכת (הסקנו שהעומס נובע ישירות מפיזור שעות העבודה בכל יום – עלייה בבקרים וירידה בערבים).

שלב 4 – התאמות בין גרפים לבון התפלגיות אמפיריות

NASA-Log File Graphs

CDF of Runtimes time – All users

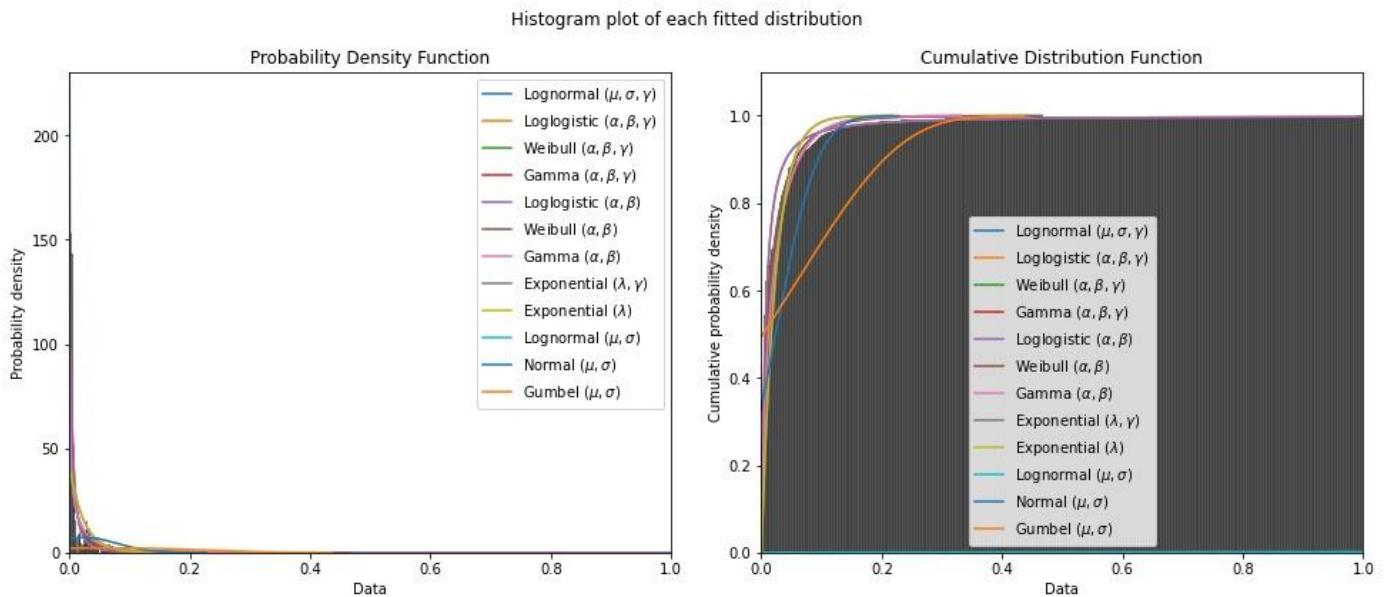


Figure 4-1

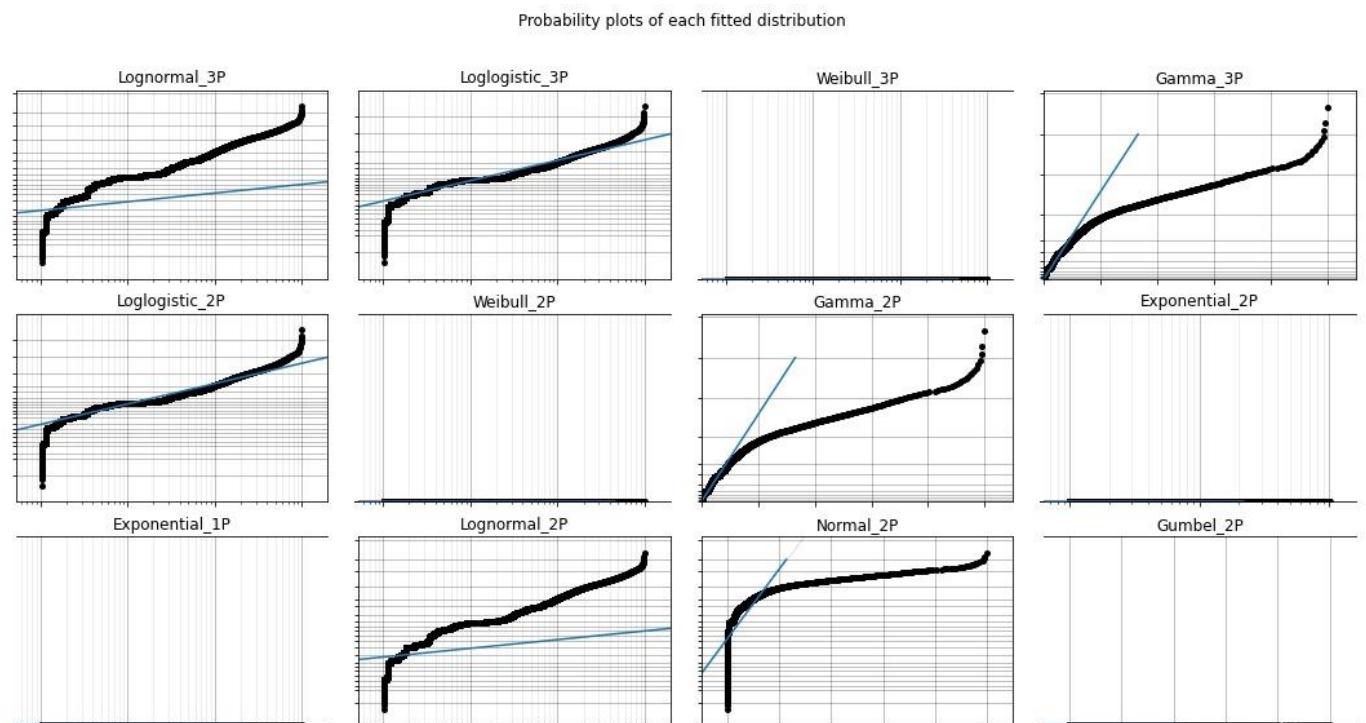


Figure 4-2

Semi-parametric Probability-Probability plots of each fitted distribution
 Parametric (x-axis) vs Non-Parametric (y-axis)

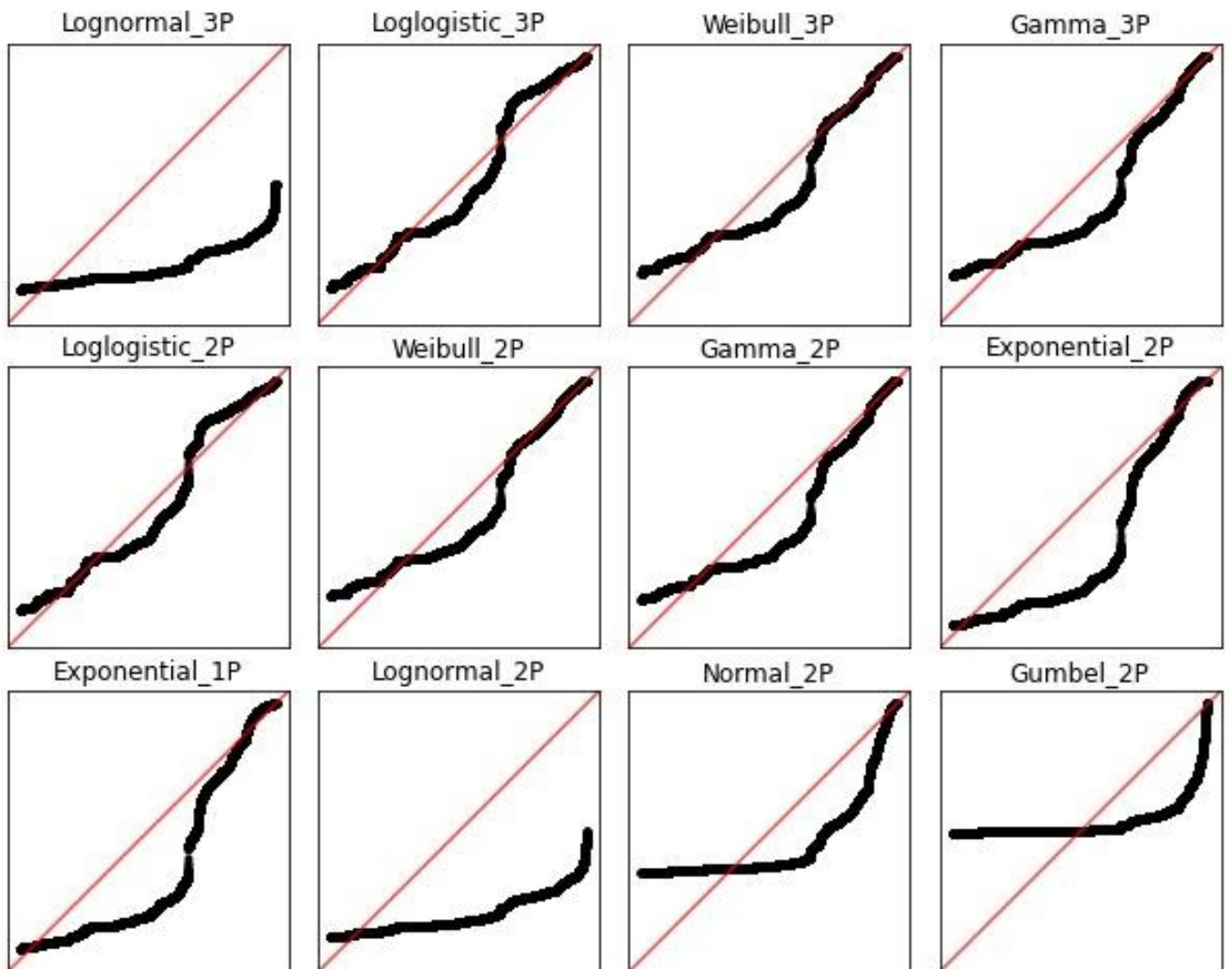


Figure 4-3

ב- CDF של ה- RUNTIMES, ובכל התת סעיפים ב-4, הסתמכנו על PP PLOTS לבודק איזה התפלגות הći מתאימה. בחלק הספציפי של RUNTIMES, מצאנו כי ההתפלגות הבי' מתאימה הייתה ה-Loglogistic, אבל גם עם איזה ההתפלגות שונה לנוב (איור 4-2 ו-4-3). ניתן לראותו התנהגות דומה בקריב ה- Weibull, Gama, Exponential, אבל ניתן להבחן כי מה שהבי' יעבוד זה Loglogistic.

CDF of Interarrival time

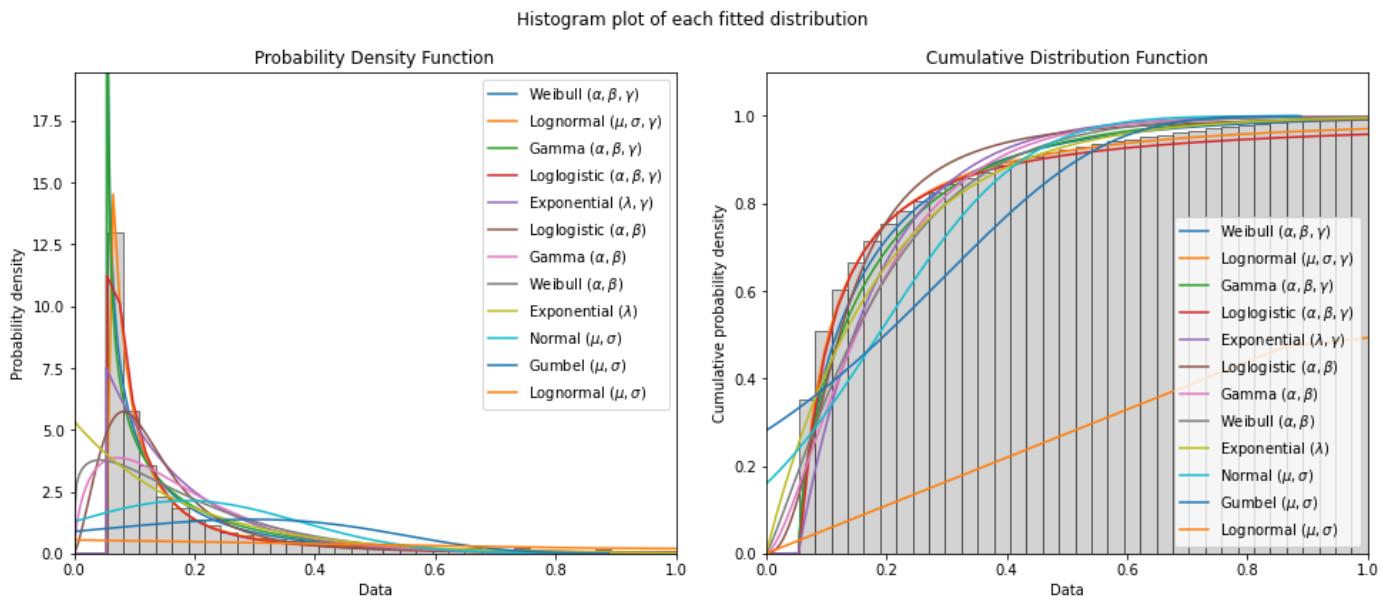


Figure 4-4

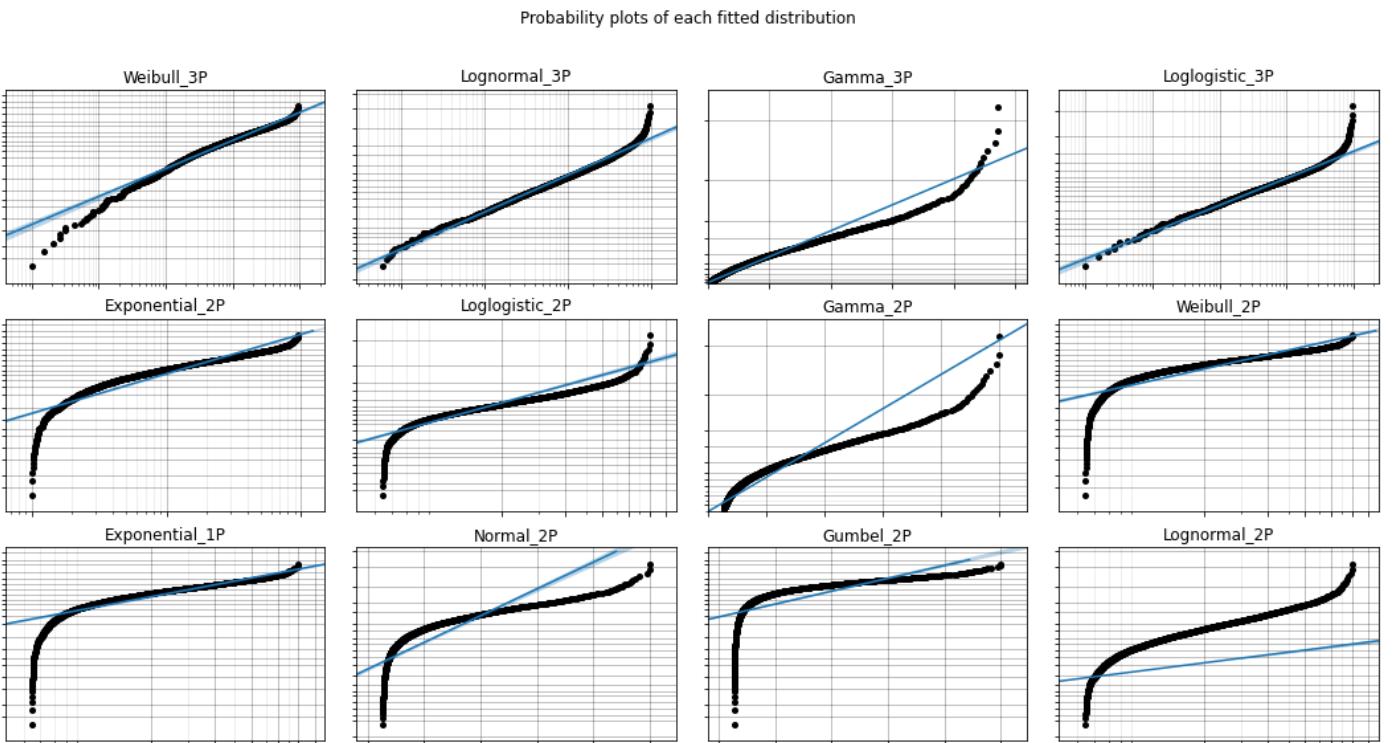


Figure 4-5

Semi-parametric Probability-Probability plots of each fitted distribution
 Parametric (x-axis) vs Non-Parametric (y-axis)

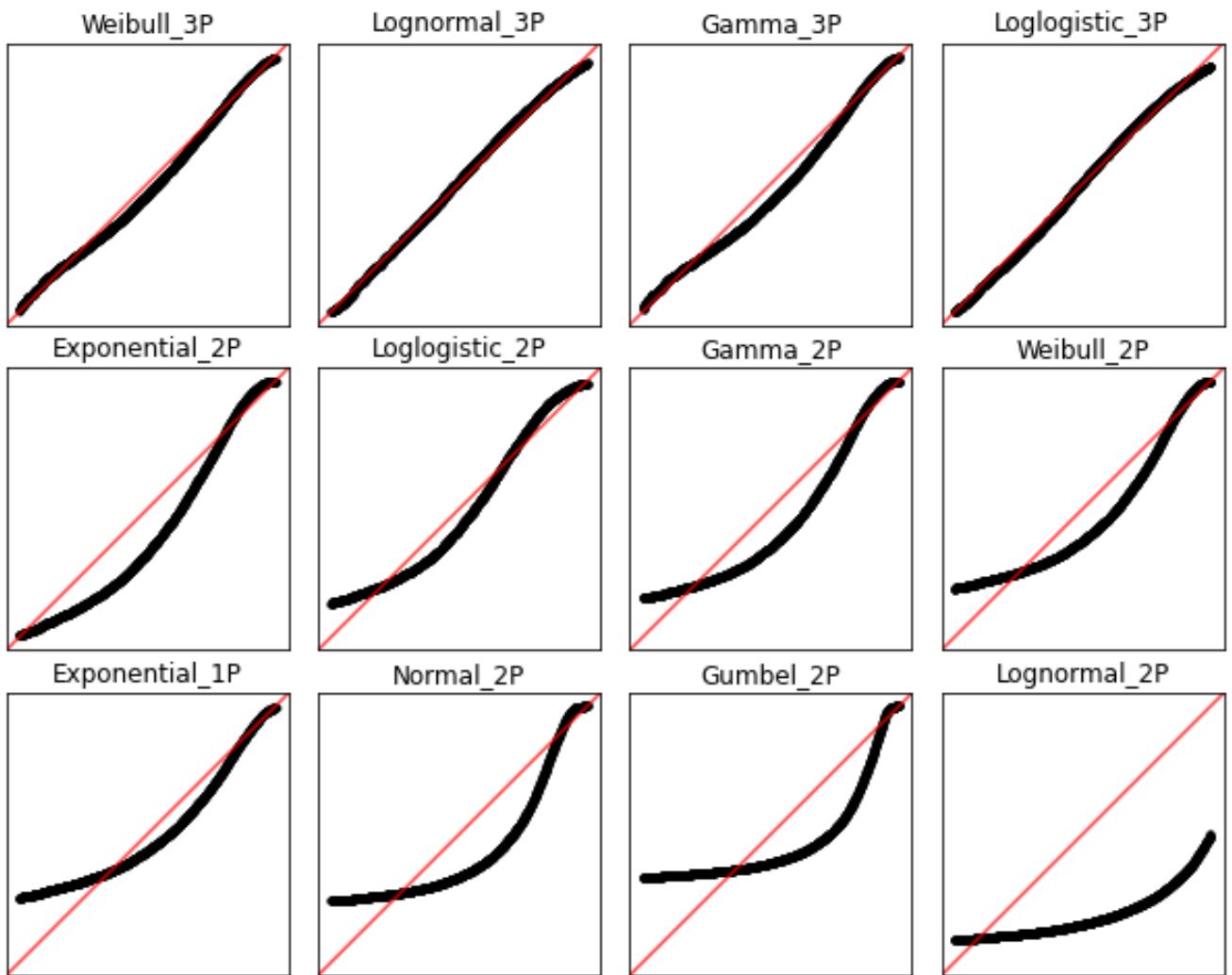


Figure 4-6

באיור 4-6 ו- 4-5, מה שהבי בולט לעין זה התפלגות 3 פרמטרים. ניתן לראות קו נקי ומסודר באיו 4-6. מאיו 5-4, ניתן להסיק כי שילוב של Lognormal או Loglogistic של 3 פרמטרים, עם Weibull יעבוד לנו טוב מאוד.

MATLAB-Log File Graphs

CDF of Runtimes

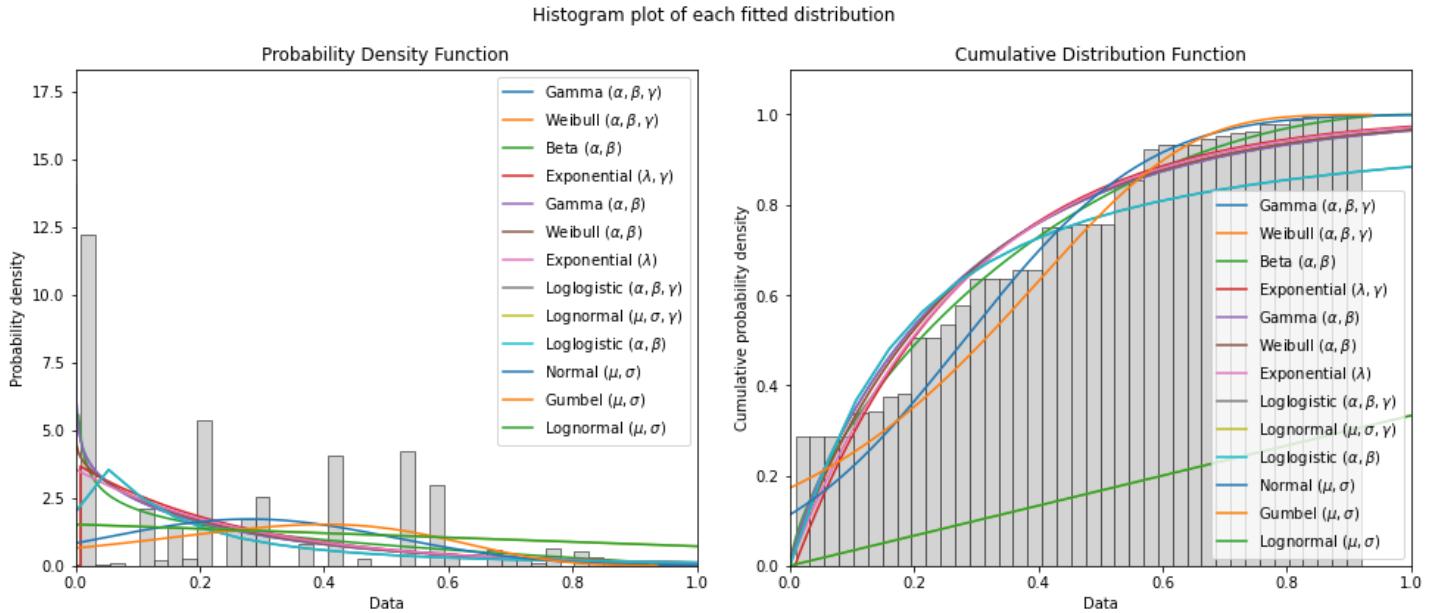


Figure 4-7

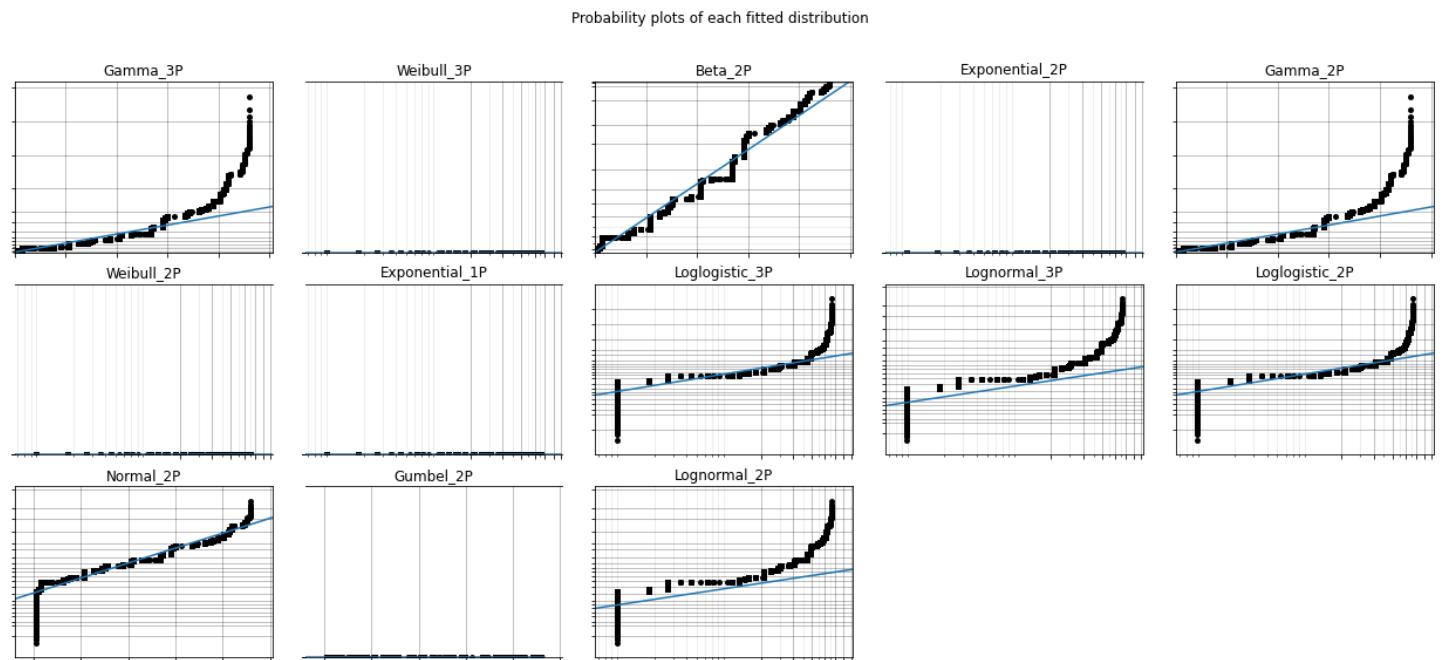


Figure 4-8

Semi-parametric Probability-Probability plots of each fitted distribution
Parametric (x-axis) vs Non-Parametric (y-axis)

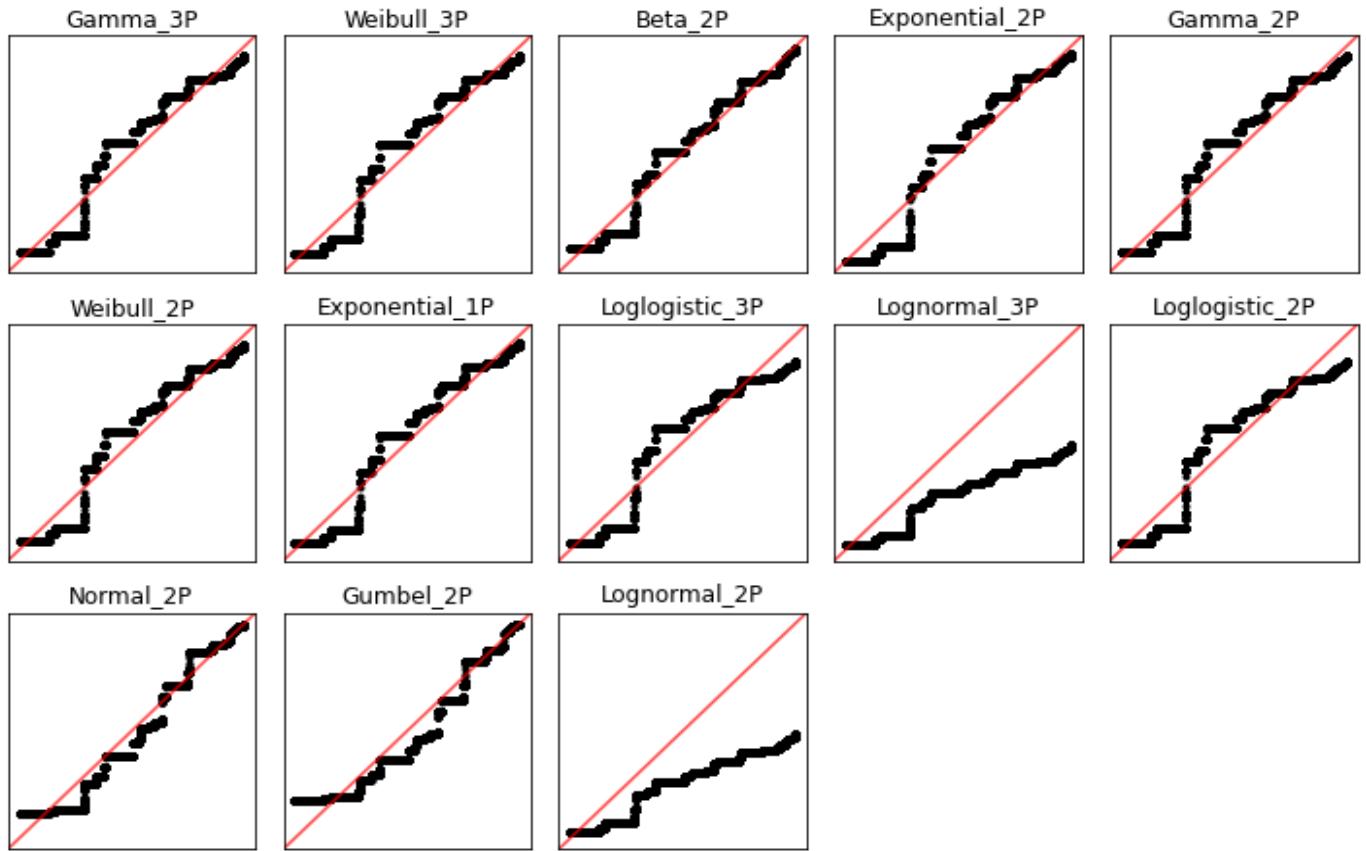


Figure 4-9

CDF of Interarrival time

Histogram plot of each fitted distribution

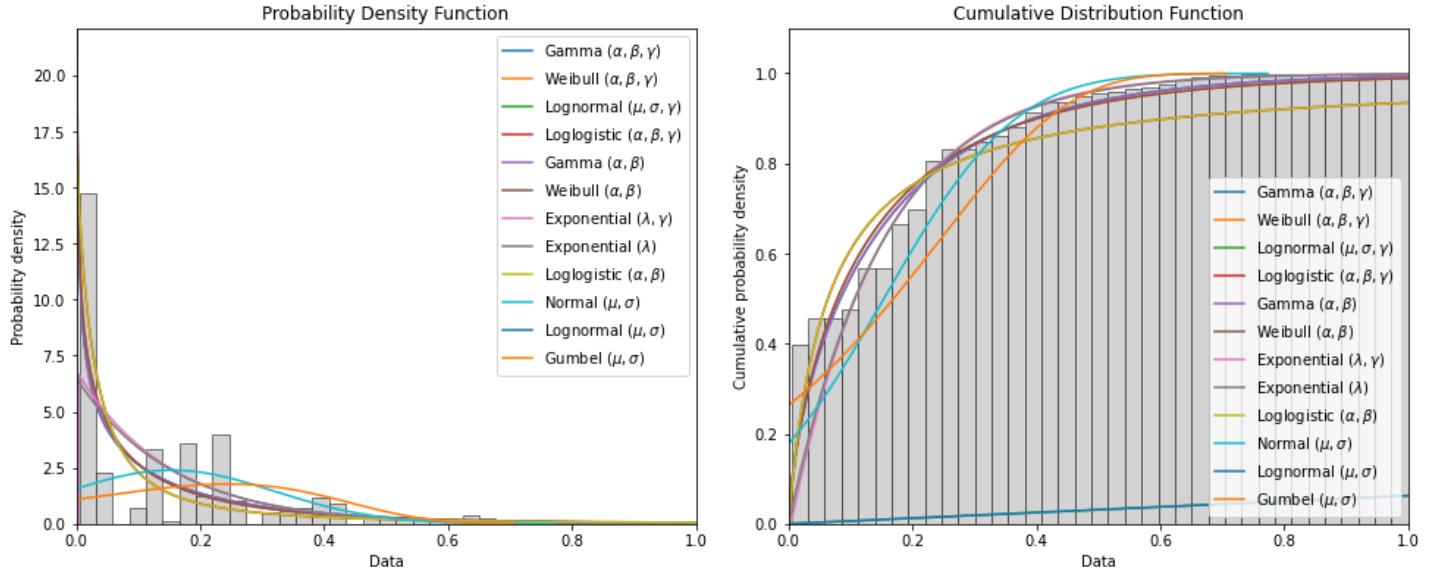


Figure 4-10

Probability plots of each fitted distribution

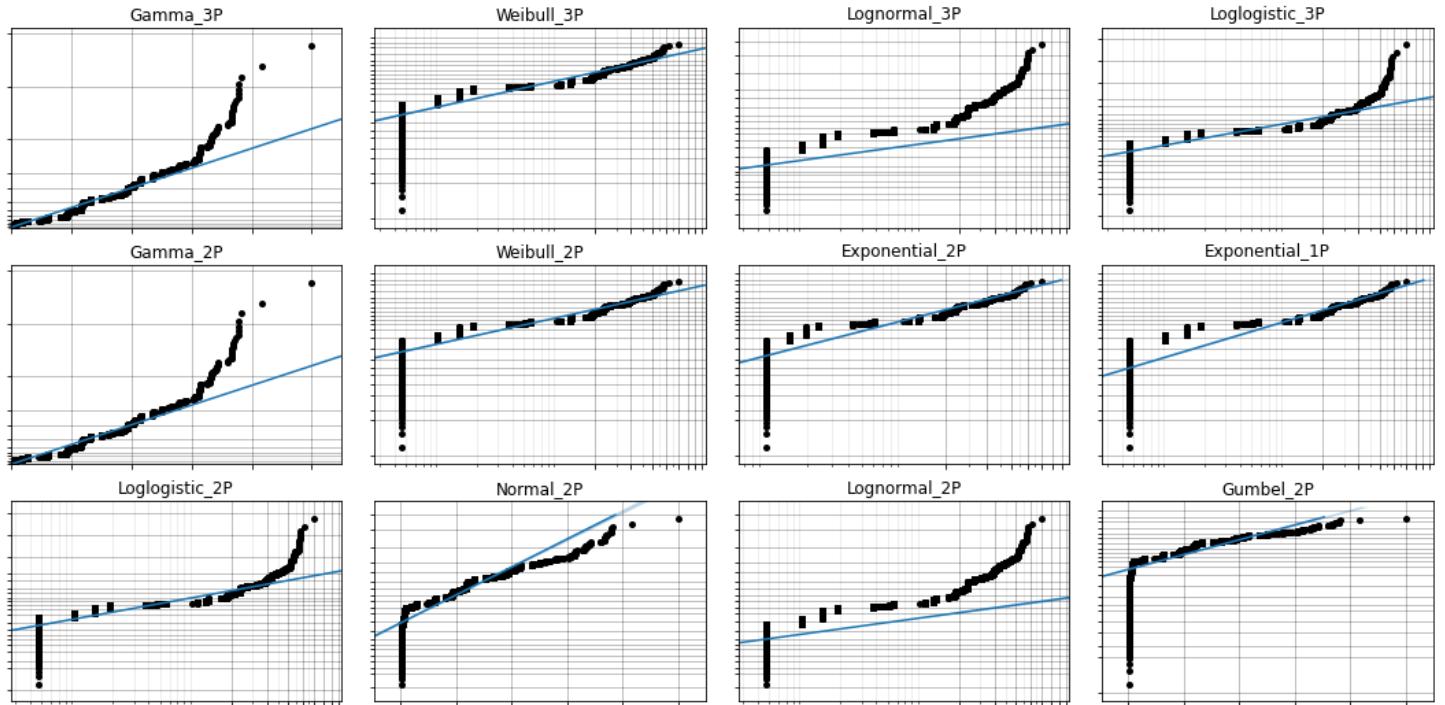


Figure 4-11

Semi-parametric Probability-Probability plots of each fitted distribution
Parametric (x-axis) vs Non-Parametric (y-axis)

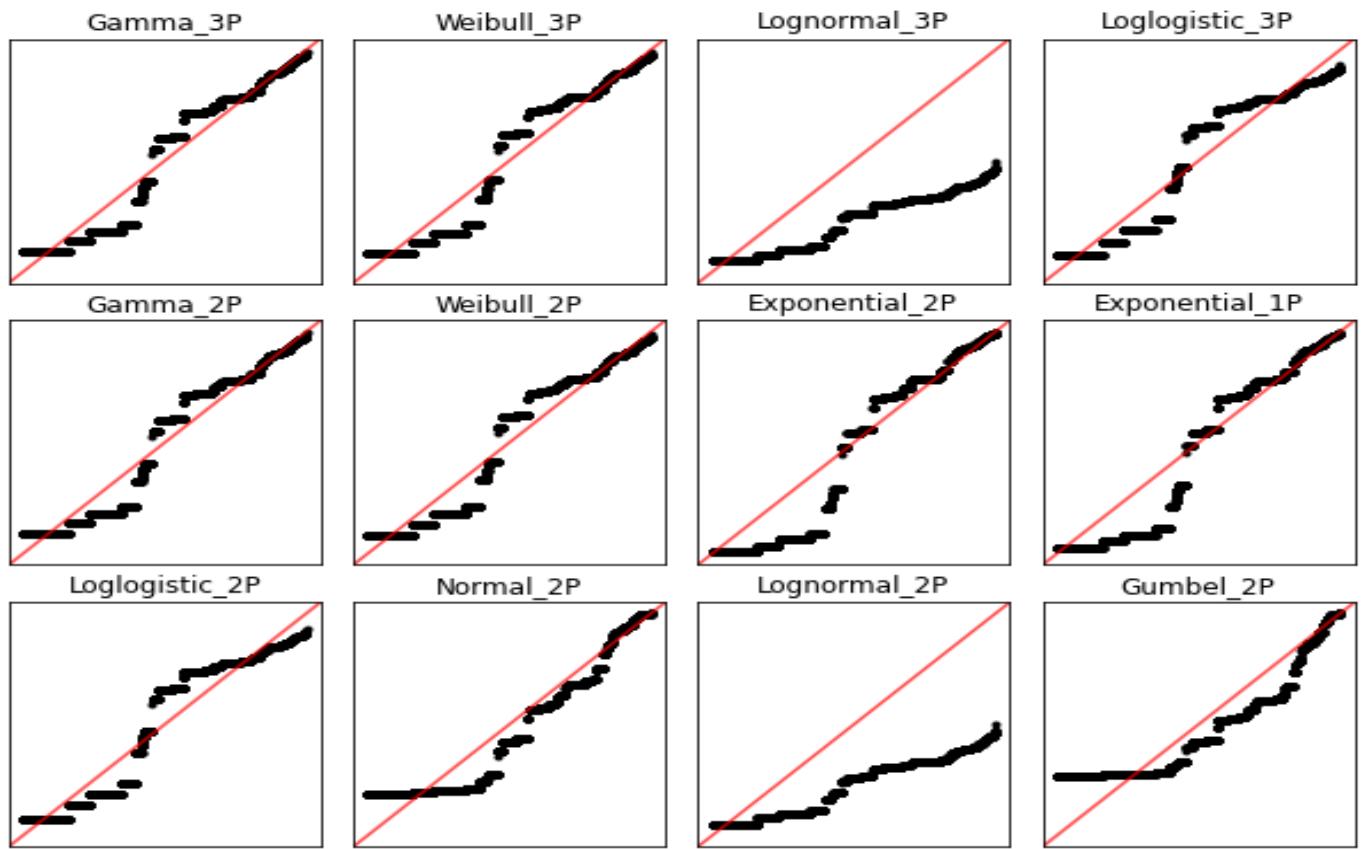


Figure 4-12

שלב 5 – התפלגות Zipf

נראה ש-2-6 figure (במota jobs לכל user במערכת) יכול להיות מתואר לפי התפלגות Zipf.

בדומה לגרף Zipf של הסתבות הופעות מילימ בקטע טקסט, ביצר גרף לפי אותו אופן. באמצעות גרפ מסוף 6 אשר מתאר לנו את במות ה-jobs לכל user, נוכל למצוא את ההסתבות להופעת job של user ספציפי.

מציג זאת בגרף:

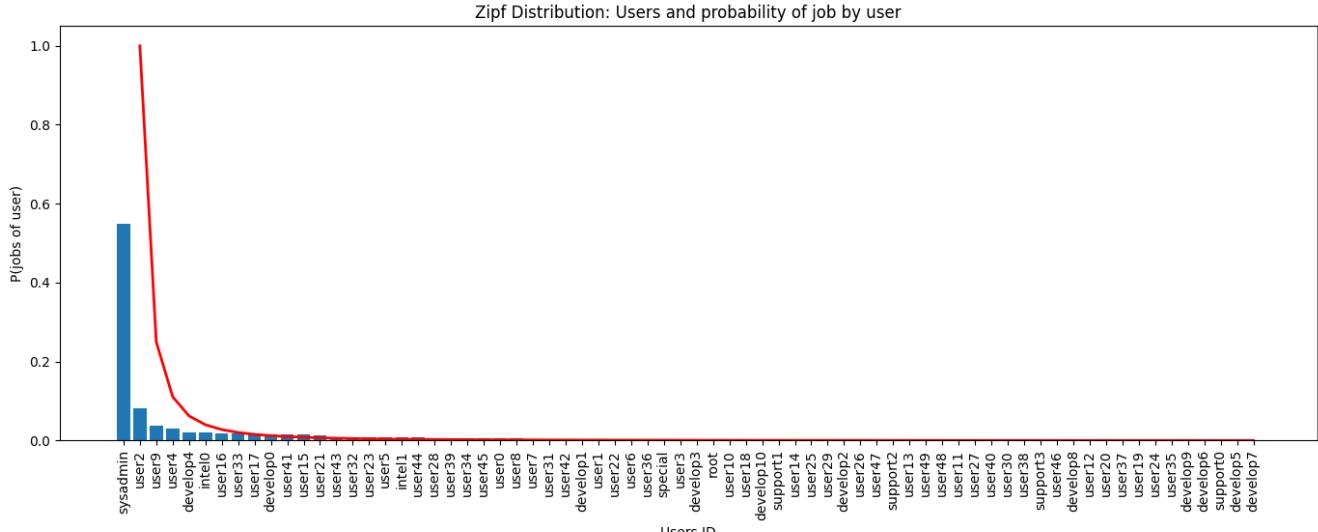


Figure 5-1

ציר ה-x יהיה המספר הסידורי של ה-users, וציר ה-y יהיה ההסתבות להופעת job מאותו user. הקן האודם מתאר את הפונקציה שנוצרת משימוש בהתרפלגות Zipf על הערכים הנ"ל.

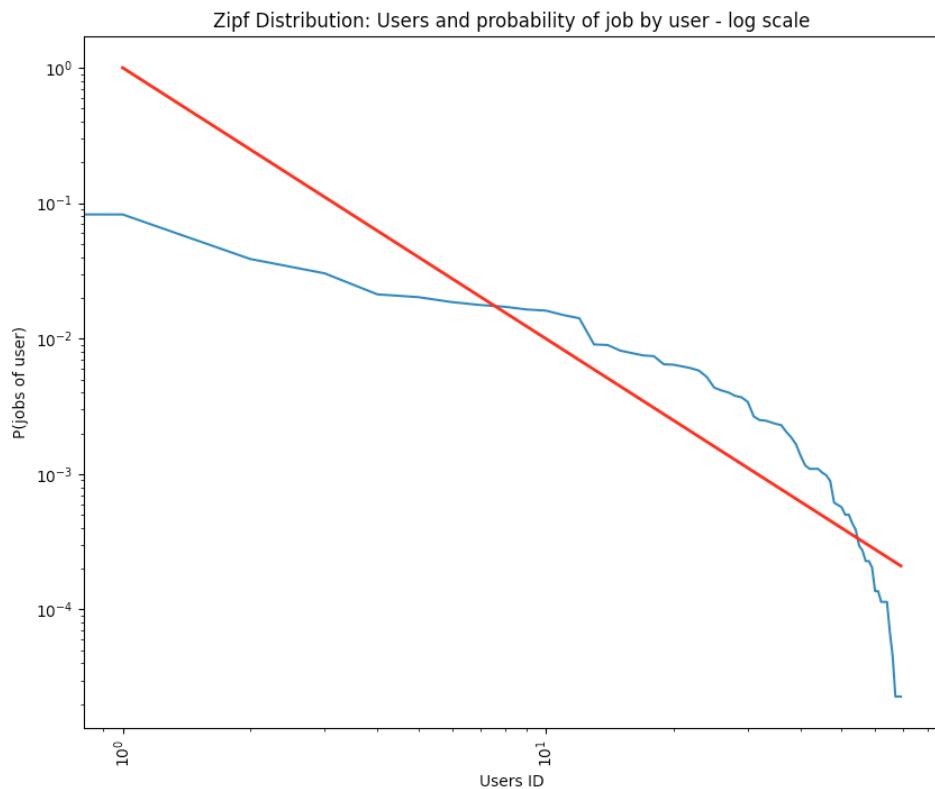


Figure 5-2

בשביל לראות יותר בנוחות את הגרף, נהפוך את העמודות לקו רציף ומייצג את צירים בסקללה לוגריתמית: ניתן לראות שההתאמה לא טוביה כפי שציפינו. ננסה לקבץ מספר דוגמאות ייחדי בשוביל התאמה טוביה יותר. בגרף הבא, נעשה קיבוץ על 5 דוגמאות. ניתן לראות שישנו שיפור כל.

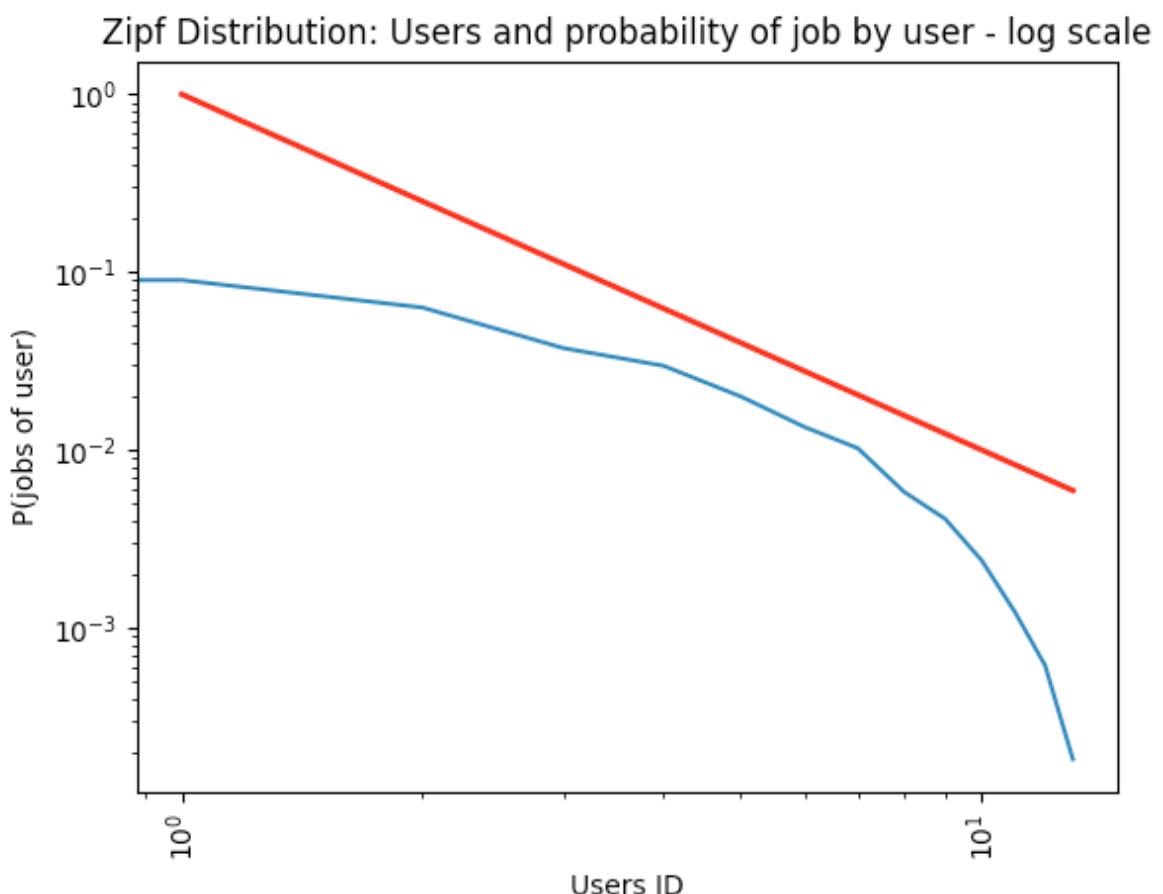


Figure 5-3

נססה לקבץ כמהות גדולה יותר של דוגימות – הפעם נקבע 7 דוגימות ונראה את הגרף הבא:

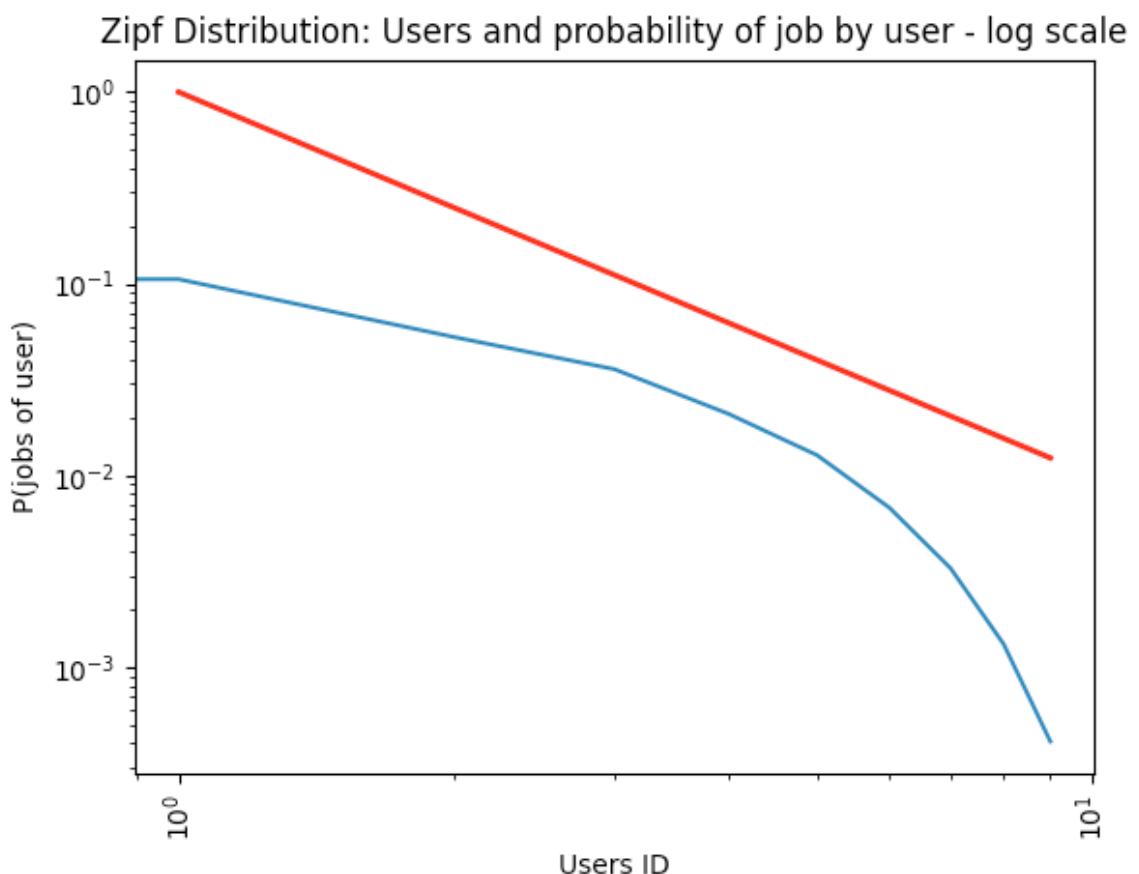


Figure 5-4

נשים לב שה-fitting בgraf עם קיבוץ של 7 דוגימות יותר מתאים לפונקציית Zipf.

שלב 6

NASA-Log File

גרפים 1-6 מייצגים את ההיסט של RUNTIMES בציר האופקי. גרף 1-6 מייצג את החלק השלילי של ה-RUNTIMES, וגרף 2-6 משלימים.

Figure 6-1: Negative think times

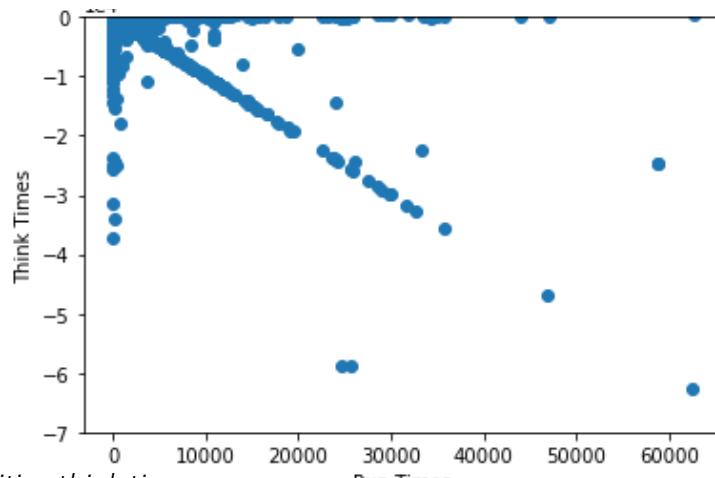
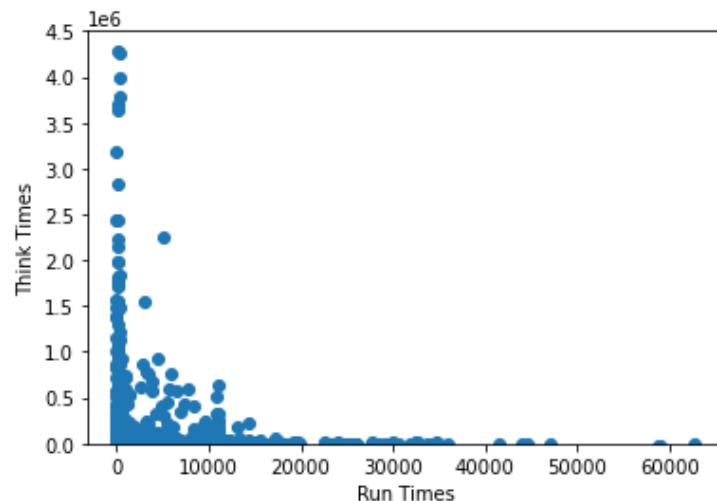


Figure 6-2: Positive think time

can see that most of the jobs have run time less than 12,000 (less or more).

On the other hand, users in NASA don't like to wait until the previous job is completed and they start to run in parallel with the previous jobs, as a result, we get that its not a must that longer jobs have longer interarrival time.

This can be also noticed in the CDF graph of interarrival times.



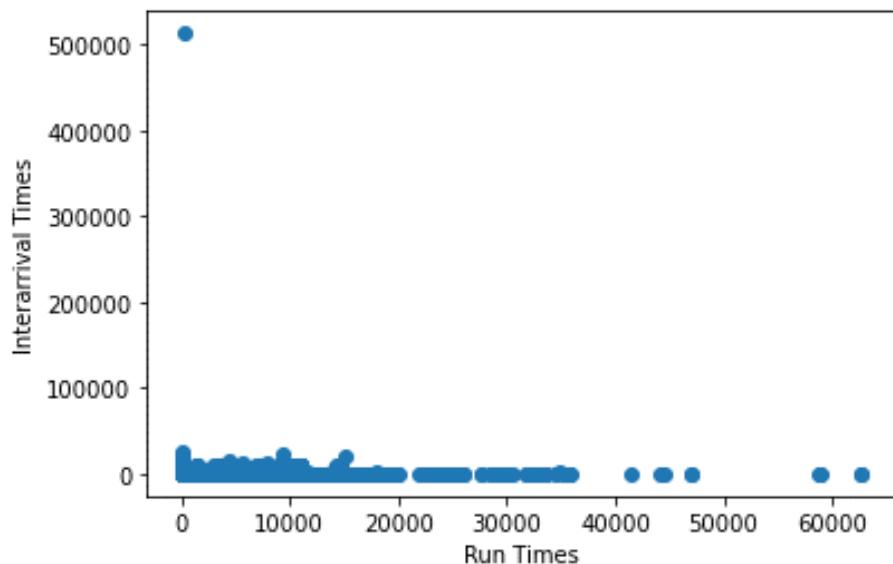


Figure 6-3: at 500000 an outlier can be observed.

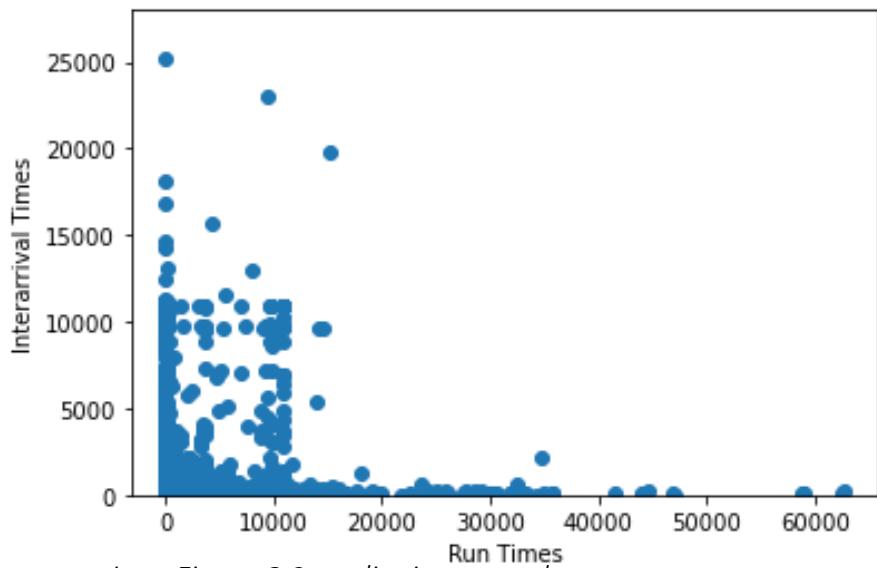


Figure 6-4: a zoom in at Figure 6-3, outlier is removed.

גרף 6-3 מתאר את ה-SWFLINTERVAL, אך ניתן להבחין ב-INTERARRIVAL שוכן נמצא בסביבות 500,000. מ

```
In [3]: runfile('C:/Users/elias/OneDrive/Documents/GitHub/SWF-Parser/main_parser.py', wdir='C:/Users/elias/OneDrive/Documents/GitHub/SWF-Parser')
Reloaded modules: RowClass
Correlation Coefficient between run times and think times:
0.0133
Correlation Coefficient between job sizes and run times:
0.1997
Correlation Coefficient between run times and interarrivals times:
0.0211

In [4]: |
```

Figure 6-5

הסתם, הנקודה הזאת יוצאה מן הכלל, שכן בנוינו את הגרף עם זום אין והציגנו אותו באור-4-6.

MATLAB-Log File

Run Times – Think Times Correlation Coefficient: -0.06555923348325185

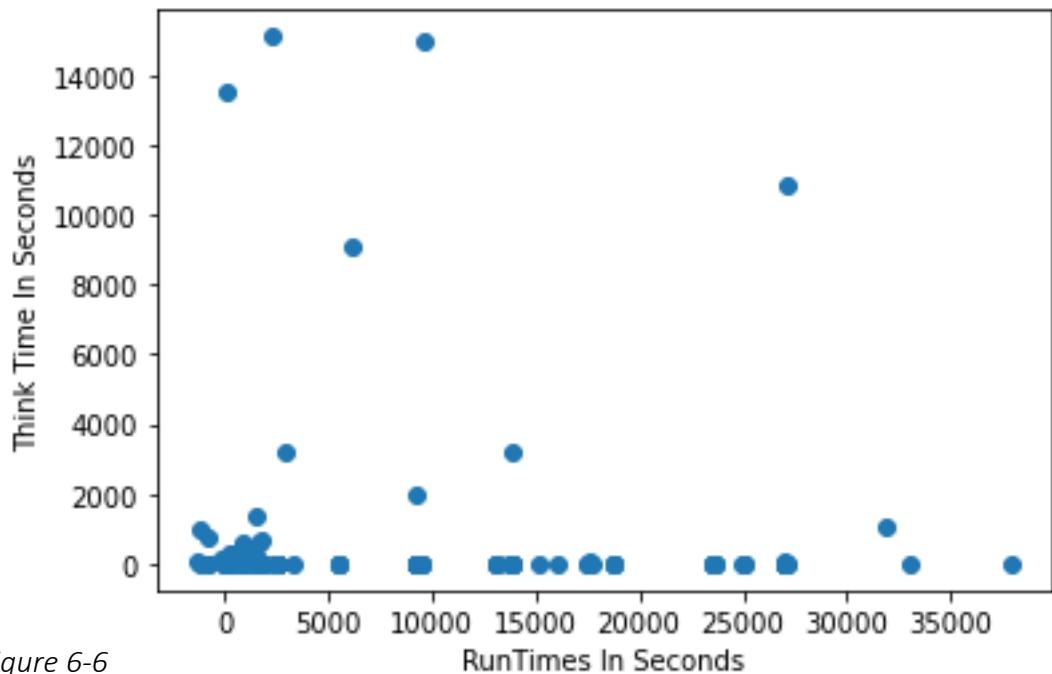


Figure 6-6

שלב 7 – יצירת distribution Hyper

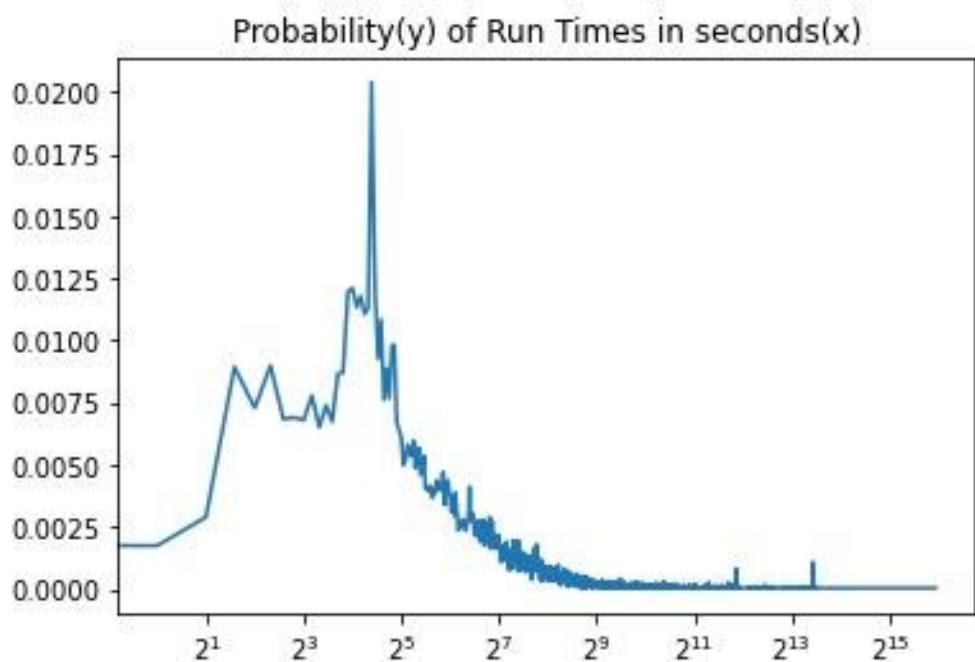


Figure 7-1

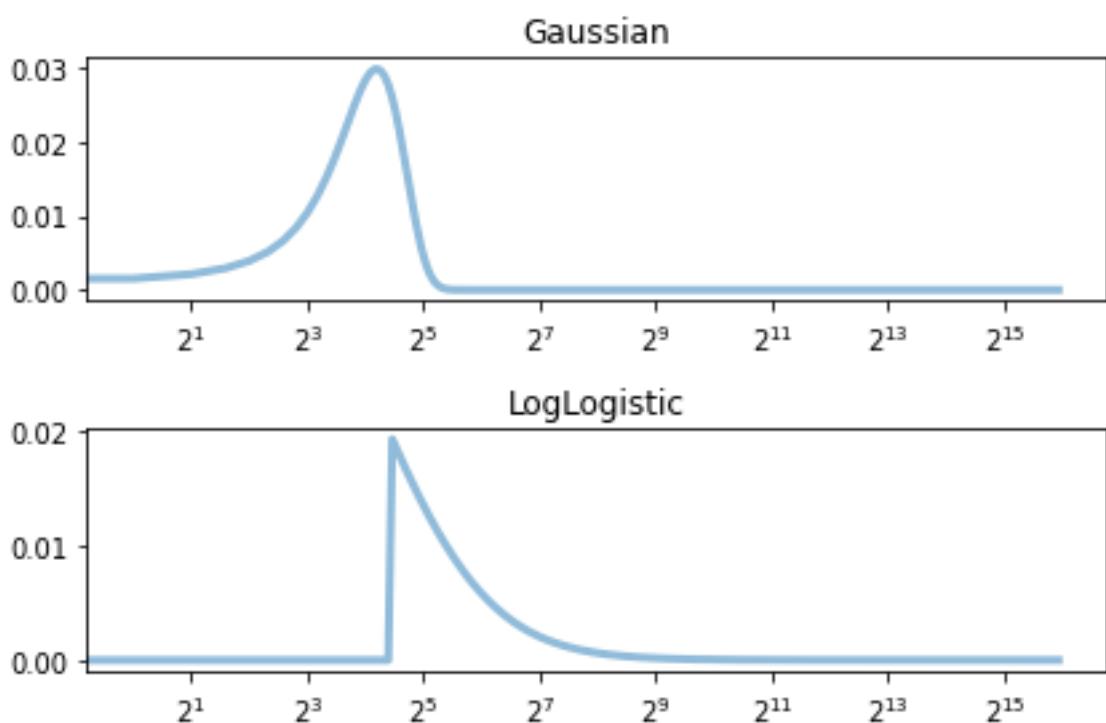


Figure 7-2

ראינו כנוכן, להשתמש בשתי ההתפלגיות המוצינות באIOR-7, ובר בנו את הsolution.

The two distributions

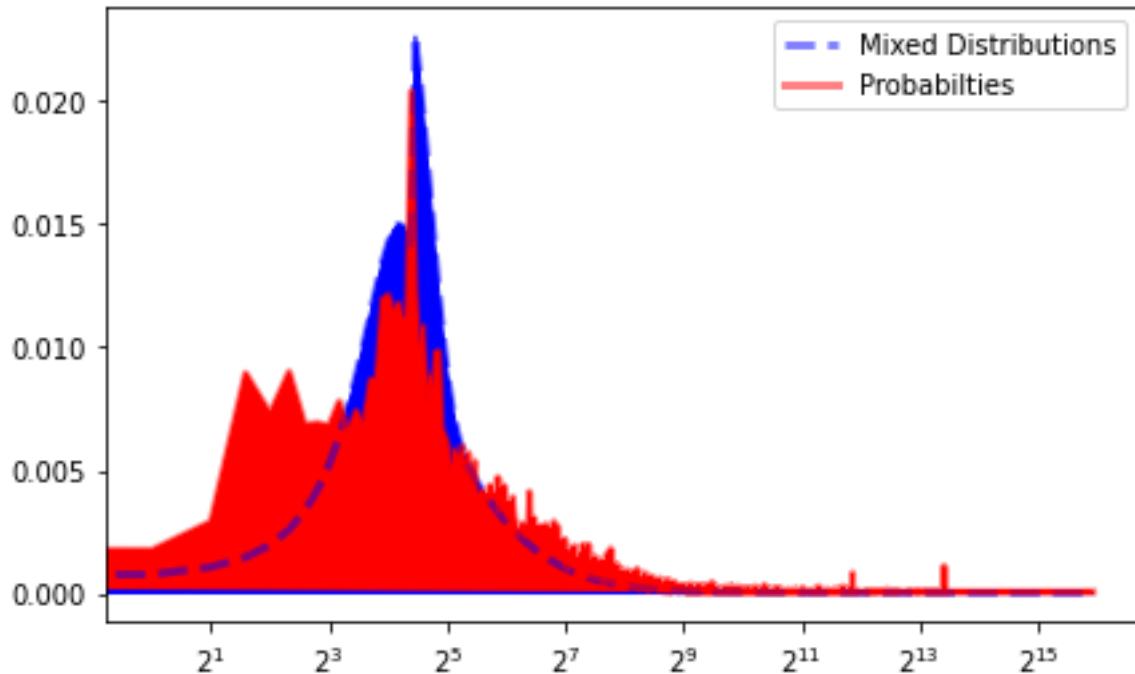


Figure 7-3

בשלב זה, השתמשנו בשיטה של EXPECTATION MAXIMIZATION, בטלב, כך בכל איטרציה בדקנו את המרחק בין הפרמטרים של ההסתברויות שקיבלנו בריצה האחורונה, לבין הרצה הנוכחית, עד שההפרש היה קטן מepsilon, שלאחר כמה ריצות קבענו על 0.0001 (הקוד של התהליך נמצא בתיקייה EM_Code_EM).

את המדידה של המרחק, היא בעצם דומה לשטח בין שני הגראפים המינימלי. באIOR-7 ניתן לראות את

```

function d = calc_distance(Param, Param_)
%
% This function calculates the distance between two sets of parameters.
Input:
    Param : old parameters
    Param_ : new parameters
Output:
    d: semi-Euclidean distance
%

d = norm(Param.mu(1) - Param_.mu(1)) + norm(Param.mu(2) - Param_.mu(2));
%
% d= sqrt((Param.mu(1,1) - Param_.mu(1,1))^2 + (Param.mu(1,2) - Param_.mu(1,2))^2 + ...
%         (Param.mu(2,1,1) - Param_.mu(2,1,1))^2 + (Param.mu(2,1,2) - Param_.mu(2,1,2))^2);
end

```

Figure 7-4: Parameters distance calculation

פונקציית מדידת המרחק. את הפרמטרים שקיבלנו, החזרנו לפייטון, ובן בנו את שתי ההתפלגיות, גאוסיאנית ולוגטיסטית.

```

function [Data_f, Param_f] = EM(Data, Param)
%
% This is the main EM algorithm. It has two steps Expectation and
% Maximization. The whole process is done in a while loop until a desired
% error has reached.
Input:
    Data: the dataset including the points and labels [x y label]
    Param: parameters of the Distributions (mu, sigma, lambda)
Output:
    Data_f: the dataset with updated labels
    Param_f: final parameters that the algorithm has converged to
}

shift = 10000; % a big number
iter = 0; % counter
epsilon = 0.0001; % percision
formatSpec = 'iteration: %d, error: %2.4f, mul: [%2.4f], mu2: [%2.4f] \n';

while shift > epsilon
    iter = iter + 1;

    % E-step
    Data_ = expectation(Data, Param);

    % M-step
    Param_ = maximization(Data_, Param);

    % calculate the distance/error from the previous set of params
    shift = calc_distance(Param, Param_);

    fprintf(formatSpec, iter, shift, Param_.mu(1), Param_.mu(2));

    Data = Data_;
    Param = Param_;

    clear Data_ Param_
end
Data_f = Data;
Param_f = Param;
end

```

Figure 7-5: Loop runs until the difference in the parameters is minimal – less than epsilon.

הפרמטרים הסופיים שקיבלנו לפי תכנית ה- EM:
gaussian= (mu=18.253,std=7.071,location=0.03)
loglogistic= (1,location=21.122,scale=50.15)

שלב 8 – ייצרת מודל המדמה את המידע הראשוני

ביצירת ה-Traces שלנו, השתמשנו בעיקרו User Resampling. עשינו זאת על ידי כך שחילקנו את ה-USERS ל-7 Clusters (השתמשנו ב-K-MEANS ובחרנו את המספר 7 משום שיש לנו 69 Users, ו-10% מהכמות הזו היא בערך 7 Users). לאחר מכן, בחנו את הקבוצות שהתקבלו, בחרנו נציגים מכל קבוצה – על מנת לדאוג שהיא לנו לפחות שתי users, Long term users והנותרים יהיו Short term users.

לאחר מכן, יצרנו קובץ קונפיגורציה בפורמט הבא:

```
"Residence      User1:<ResidenceTime1>      User2:<ResidenceTime2>      ....  
User69:<ResidenceTime69>"  
  
//For Example: Residence User1:4892 User2:12099 User3:123 .... User69:3  
  
"Activity      Week1:<NumberOfNewUsers1>      Week2:<NumberOfNewUsers2>      ....  
Week13:<NumberOfNewUsers13>"  
  
"Random_Seed 1"  
  
<Jobs of the first user chosen>  
  
.  
  
.  
  
.  
  
.  
  
.  
  
.  
  
.  
  
.  
  
<Jobs of the last user chosen>  
  
//The number of users chosen must be 2 long term users (i.e are active the whole time) and  
6-7 short-term users (you could add more if you want)
```

כשבחרנו את USERS, לא יכולנו להסתמך על השיטה של ה-KMEANS, ולבחו נציגים שקיימים CENTROIDS, משומם שהייתה לנו עד סגיה וכוונה שchipshno אותה בכל נציג מכל CLUSTER. היה לנו חשוב לבחור נציגים של LONG TERM ו-SHORT TERM, ויש CLUSTERS שקיבלו שיש בהם רק SHORT TERMS, ובאה שיש בהם רק LONG TERMS, לכן בחרנו ידנית ב-USERS לאחר שבדקנו והתייחסנו לסוגיה שהציגו. סוגיה אחרת שנטקלנו בה (בשנינו לחת לאלגוריתם לבחור נציגים), היא שנבחרו נציגים שיש להם כמה גובהים – שאפשר לספר ביד אחת, מה שגורם לכך שהטiris הכליל נתונים לא תקין, למשל שככל הטiris מכיל רק 12 גובהים והכפולות שלהם. לכן העדפנו לבחור ידנית לפי האילוצים שהציגו.

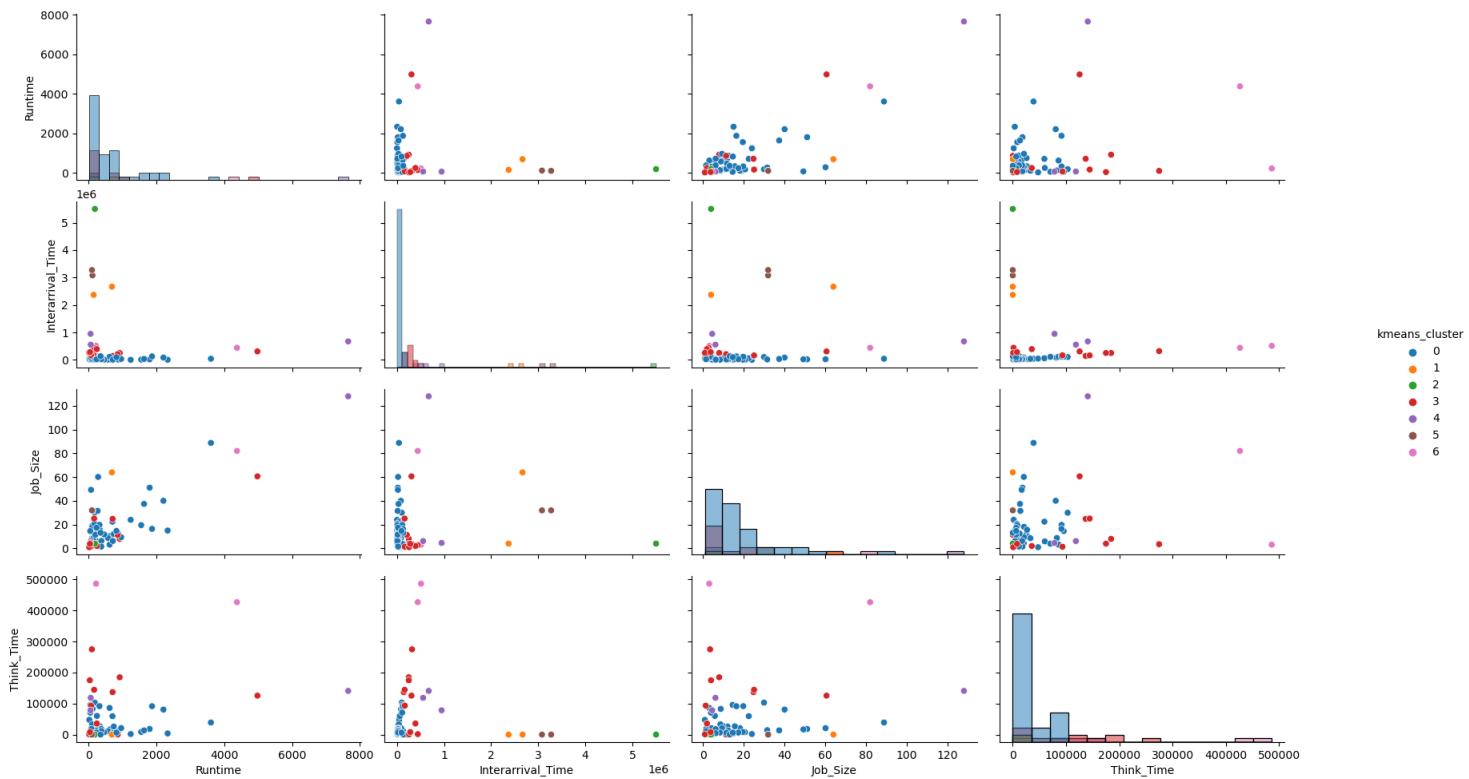


Figure 8-1: K-Means clusters with all axis combinations.

לאחר בחירת-users , יצרנו שלושה קבצי Config, אשר כל אחד מהם מגדה עומס שונה במערכת; 80% - 100% - 120%. חשוב להזכיר את הצעד הזה בשיטת User Resampling שלפיה עבדנו:

יש לנו את מספר המשתמשים החדשניים שמתווספים למערכת בכל שבוע. בכל שבוע, בטריס החדש, דוגמים מהמשתמשים שיש לנו ביד לפי מספר המשתמשים המתווספים מחדש בטריס המקורי.

לගבי יצירתה 80% Load | 120% Load | 100% Load configuration file שורה שבתוכה בה עט מספר מסויים נגיד 80 אז בתוך התוכנית שלנו :

```
# Multiply the number of new users per week with the given load to generate more load on the system
load=/100
for key in NewUsersPerWeek:
    splittednum=math.modf((NewUsersPerWeek[key]*load))
    NewUsersPerWeek[key]=int(splittednum[1])
    probability=splittednum[0]
    if np.random.random()<=probability:
        NewUsersPerWeek[key]+=1
```

אנחנו מכפילים את מספר המשתמשים החדשניים שמתווספים בכל שבוע בכמות ה Load (0.8 או 1 או 1.2 כפול מספר f זהה הפקטור המוכפל כדי להגיע למוצע העומס הדרוש (למדנו את השיטה הזאת בהרצאה الأخيرة במצגת האחרונות), ואז יש סיכוי לא קטן שנתקבל מספר לא שלם אז עבדנו לפי שיטה שלמדנו

בהרצאה האחורונה של הקורס, שמספרים את המספר לשני החלקים השלם והשבר ואז בהסתברות של השבר מוסיפים אחד למספר השלם.

בחירת השיטה של User Resampling הייתה לפי שיקול אחד ייחיד: רצינו לעבוד לפי שיטה שהיא יחסית לא קשה ובאותו זמן משוחררת לנו כמו שיותר מופיעים מהtrace המקורי.

הוספנו את קבצי הקונפיגורציה לתיקייה שלנו, ובתוכננות Python, בנוינו את שיטת ה-User Resampling אשר בהתבסס עליה הפקנו את הגրפים שנראוה בשלב 9.

ציפיות:

- היציפה שלנו, שסעיפים 1 ו 2 משלב 9 (Consumption, Runtimes, Interarrivaltimes & User distribution) יהיו הבסיס הטריוויאלי של התוצאות שנתקבל, כי הם אבני בסיס של הטריס ושל השחרור שלו.
- לפי השיטה שלנו, שמסתמכת על דוגמה משתמשים מהטריס המקורי, נבע מכך שאפשר לשחרר את הקורלציה בין מאפיינים שונים, SELF SIMILARITY - LOCALITY OF SAMPLING. כמו כן, הנציגים שנבחרו,אפשרים לנו לשמר גם על המחרזר היומי והשבועי של הטריס.
- לא ציפינו שנוכל להציג את ה RATE SUBMISSION של היוזרים כולם מהטריס המקורי, מהסיבה הפשוטה שאנו מסתמכים על 7 יוזרים מתוך 69. גם לאותה סיבה, לא ציפינו שנוכל לשחרר טרנדים.
- בטריס המקורי לא היה לנו WAITTIMES ולכן לא ציפינו שנחזר.

שלב 9- אימות

.1 Consumption

בגלל הצפיפות של זמן אי אפשר לראות את התוצאות שלנו בצורה טובה, לצורך כך חישבנו את אחוז העומס הממוצע בכל רגע זמן נתון לאורק כל המערכת. מצטבר ש:

הלוג המקורי שלנו היה בעומס של 49.19% ז"א שבממוצע ~ 64 משבץ היה תפוסים לאורק כל המערכת. לעומת זאת בטריזיסים של ה- 80% העומס קיבלנו את המספרים: 35.45%, 38.15%, 43.17%, 45.29% ו- 39.26% אז כאן הגענו להתוצאה טוביה (עם השמירה על שאר תכונות הטריזיס).

בטריזיסים של ה- 100% העומס קיבלנו: 48.93%, 44.17%, 45.29%. הבעיה הייתה בטריזיסים של ה- 120% העומס שmmoוצע אחד העומס עלה אבל לא ב- 20% : 54.34%, 53.44%, 51.20%.

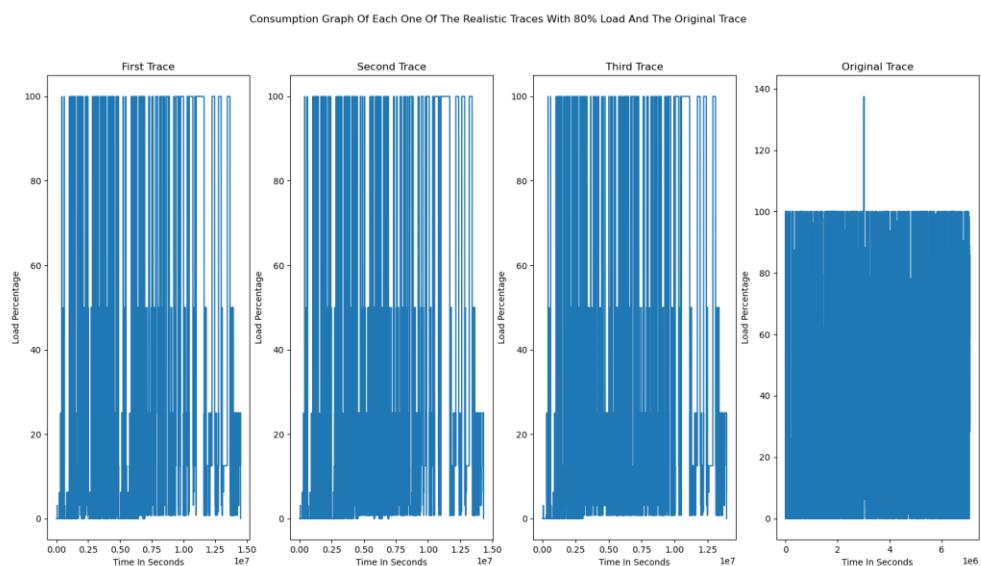


Figure 9-1: Consumption graphs of 80% load.

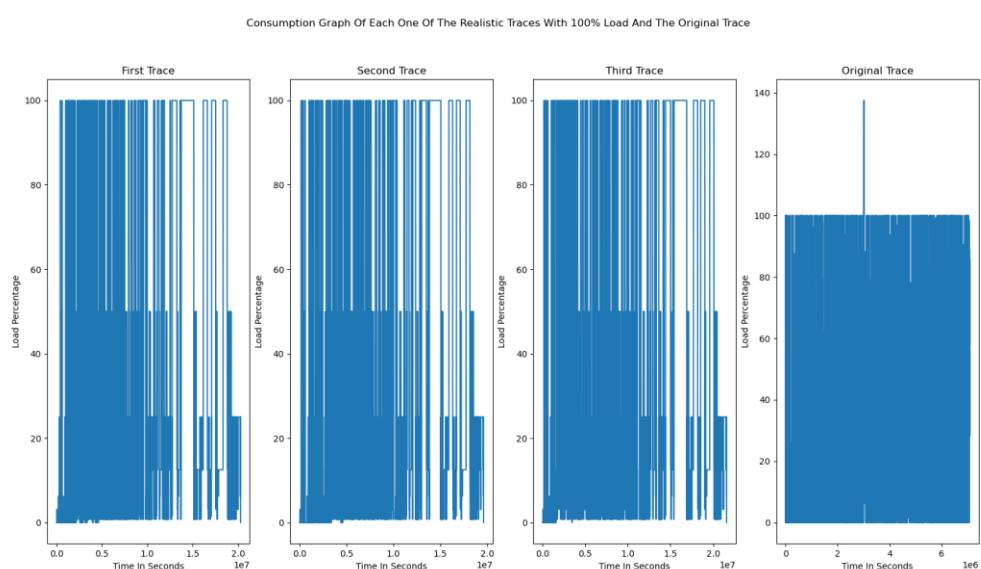


Figure 9-2: Consumption graphs of 100% load

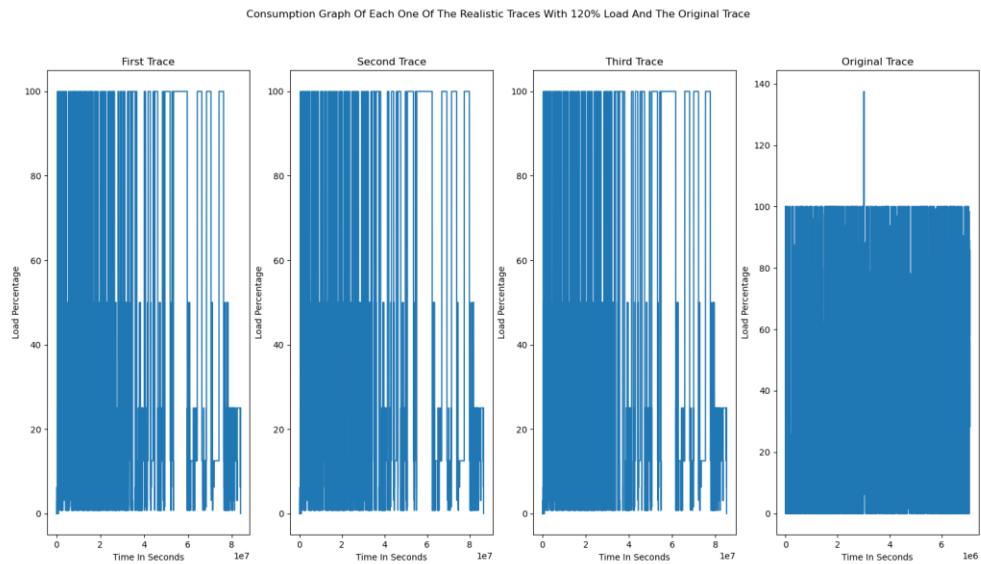


Figure 9-3: Consumption graphs of 120% load

2. התפלגיות Interarrival time .1

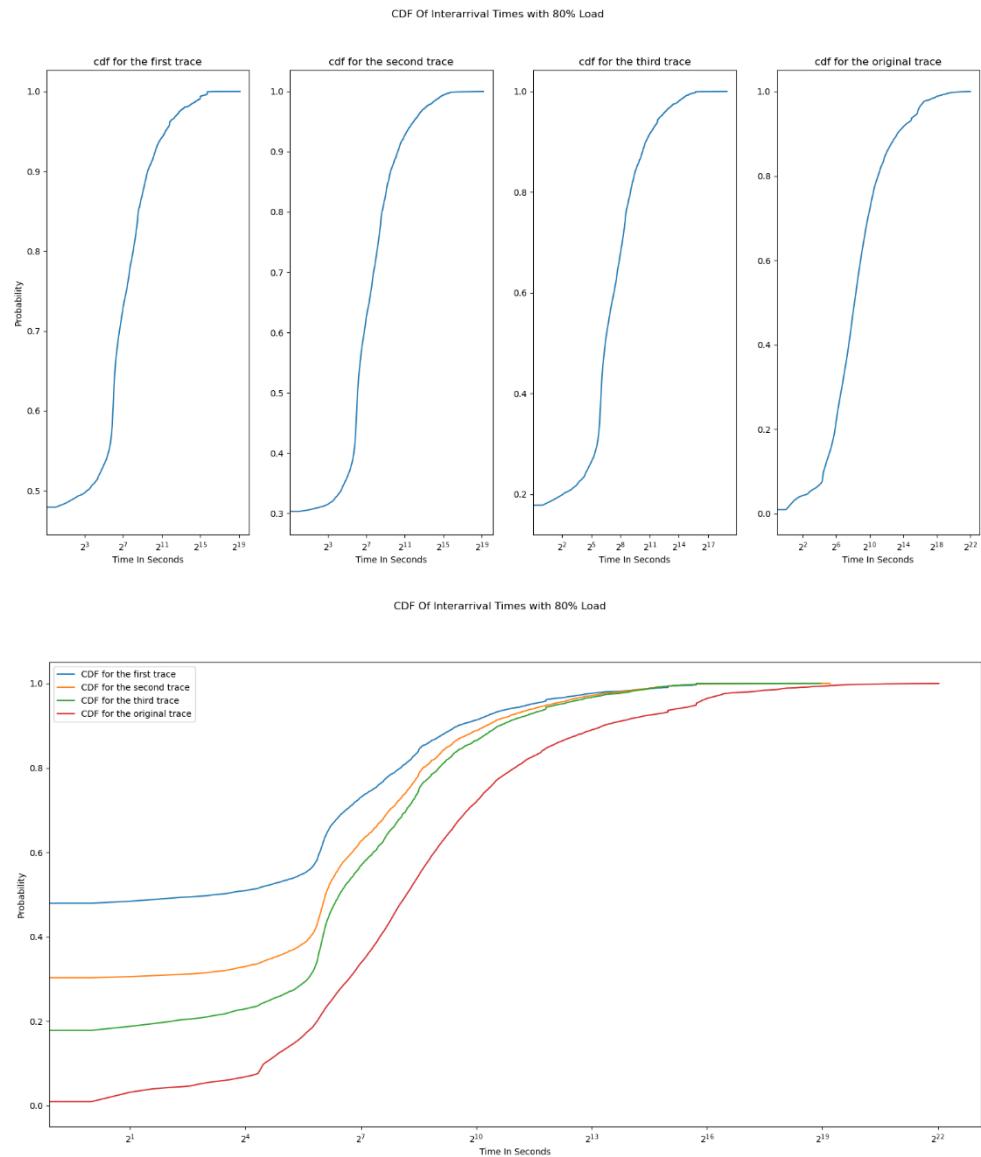
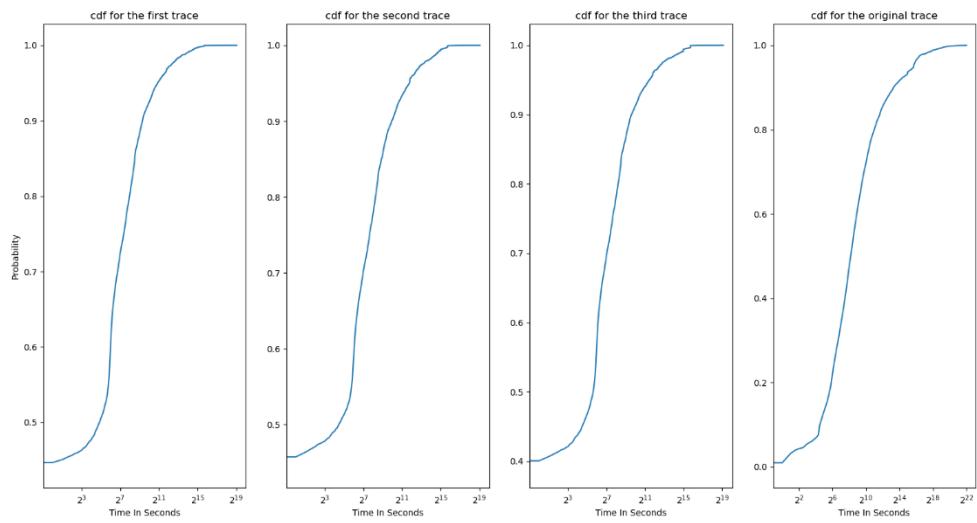


Figure 9-4: CDF of interarrival times with 80% load

CDF Of Interarrival Times with 100% Load



CDF Of Interarrival Times with 100% Load

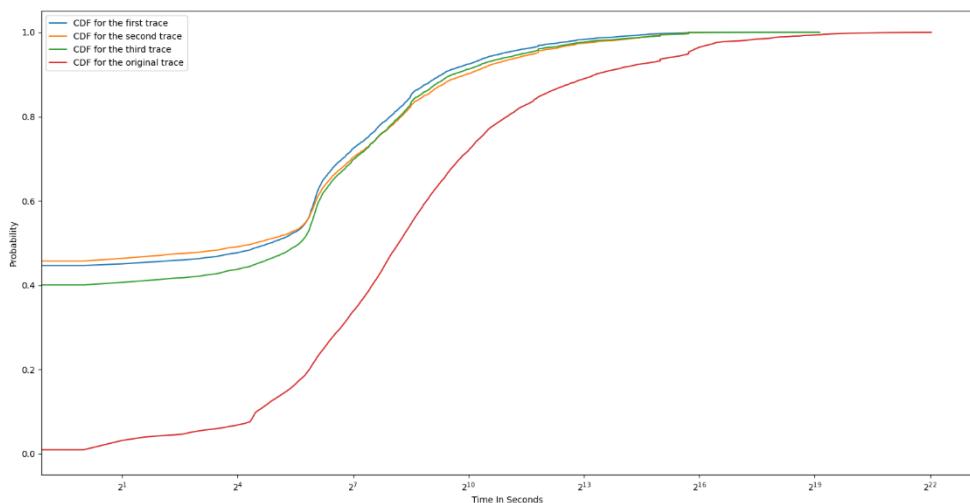


Figure 9-5: CDF of interarrival times with 100% load

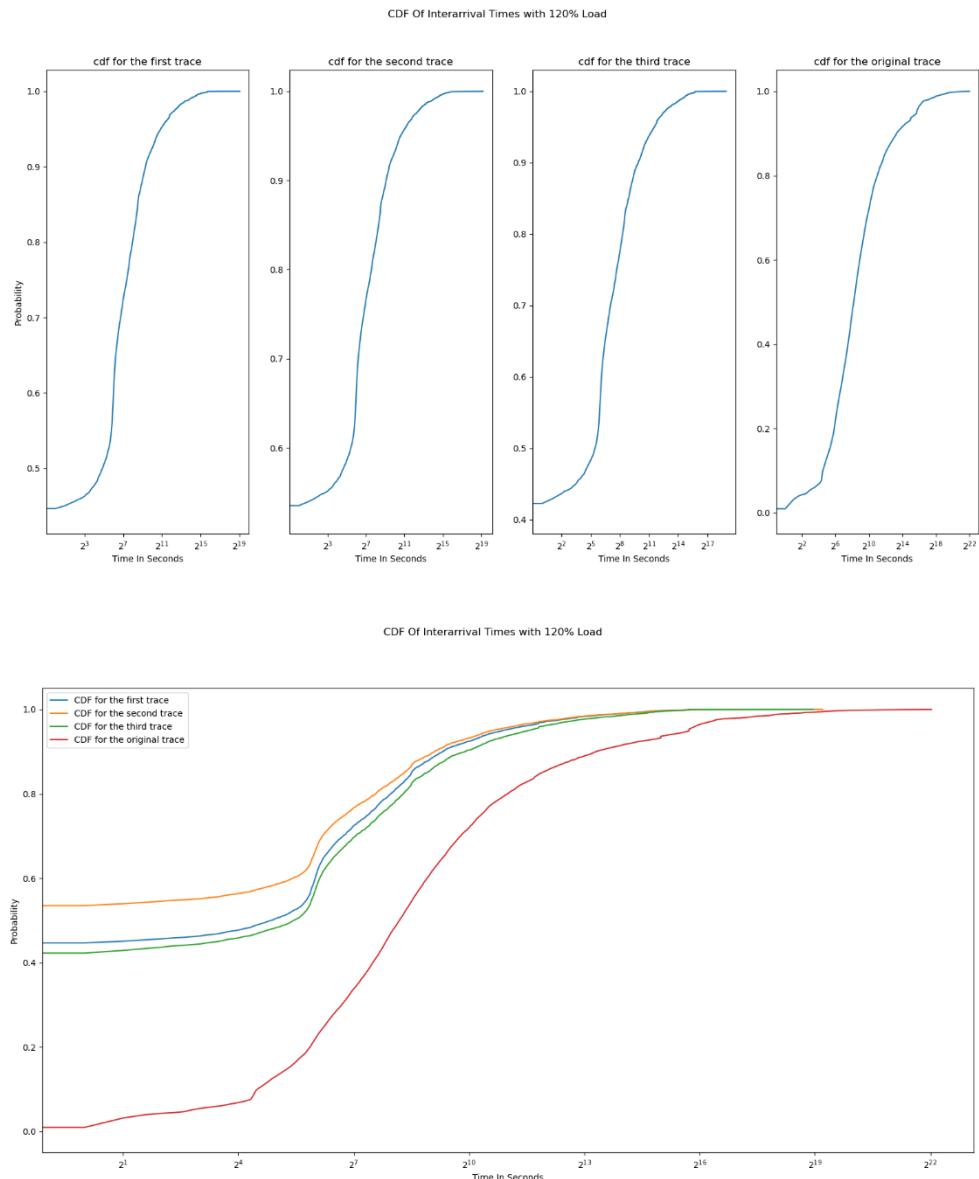


Figure 9-6 CDF of interarrival times with 120% load

אנחנו מראים בכל אחד מהאיורים את פונקציית CDF בסקלת log ביציר האופקי מה שבעצם מאפשר לנו לראות שזמני הריצה מתפלגים באופן ממש דומה בכל הטרויסים, מה שנובע מבחירה טובה ב clustering techniques שהתקיים בשלב הקודם. הצלחנו לבסות את רוח זמני הריצה הקיימים במערכת, לעומת זאת בגרפים של ה interarrival times יש קצת שינוי בCDF של הטרויס המקורי לעומת הטרויסים שנוצרו, זה נובע ממהות השיטה של - User Resampling - כי מרחיב הדגימה שלנו מכל 9 משתמשים שהם דוגמים לאורך כל הטרויס אז הסטלים חוזרים על עצם שוב ושוב וכך נוצר השוני הזה. אולי אם היינו מוסיפים עוד משתמשים למרחב הדגימה שלנו או הייתה לנו בחירה אחרת של משתמשים היינו יכולים לשחרר את התפלגות ה interarrival times יותר טוב.

:Runtimes .2

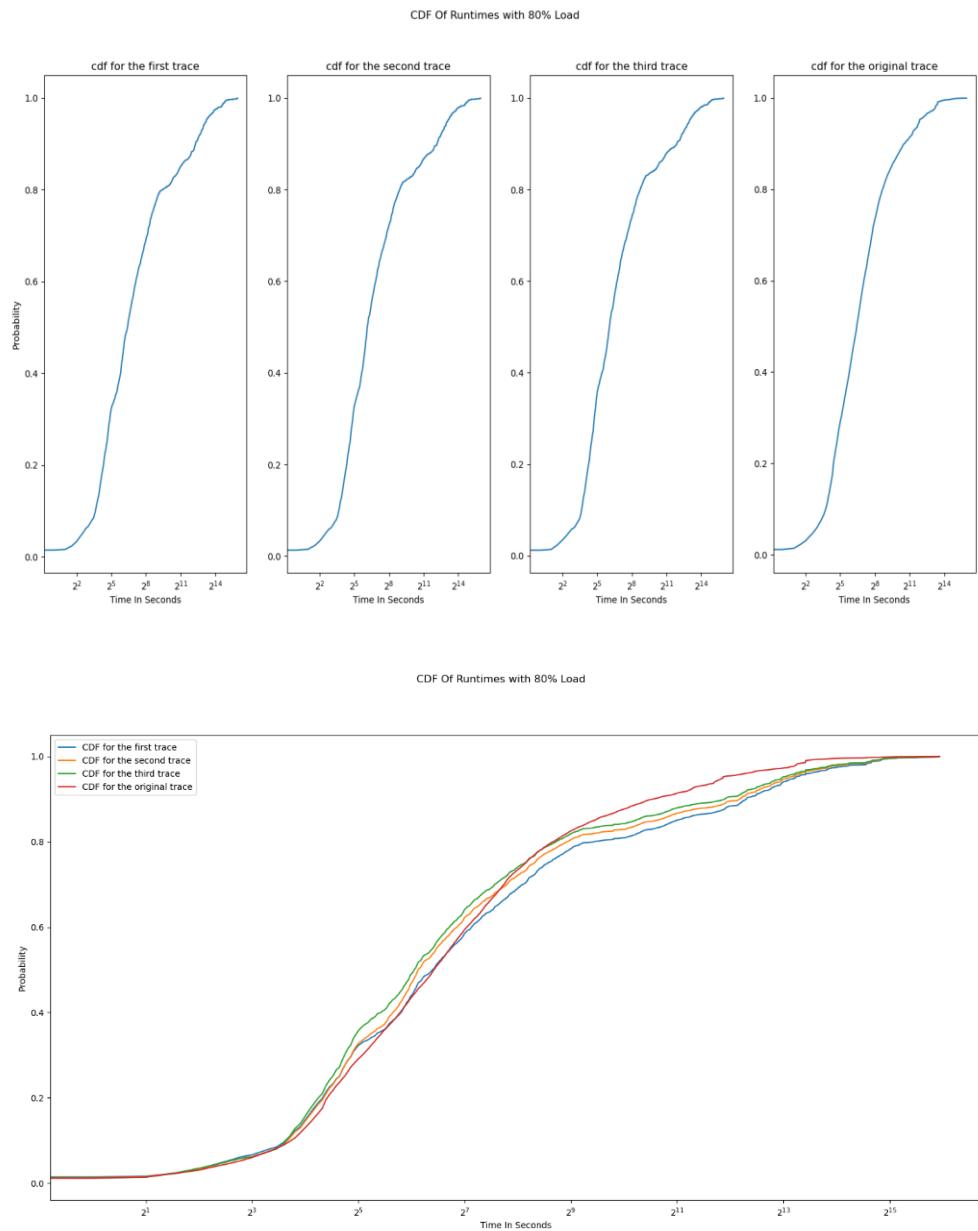


Figure 9-7: CDF of runtimes with 80% load

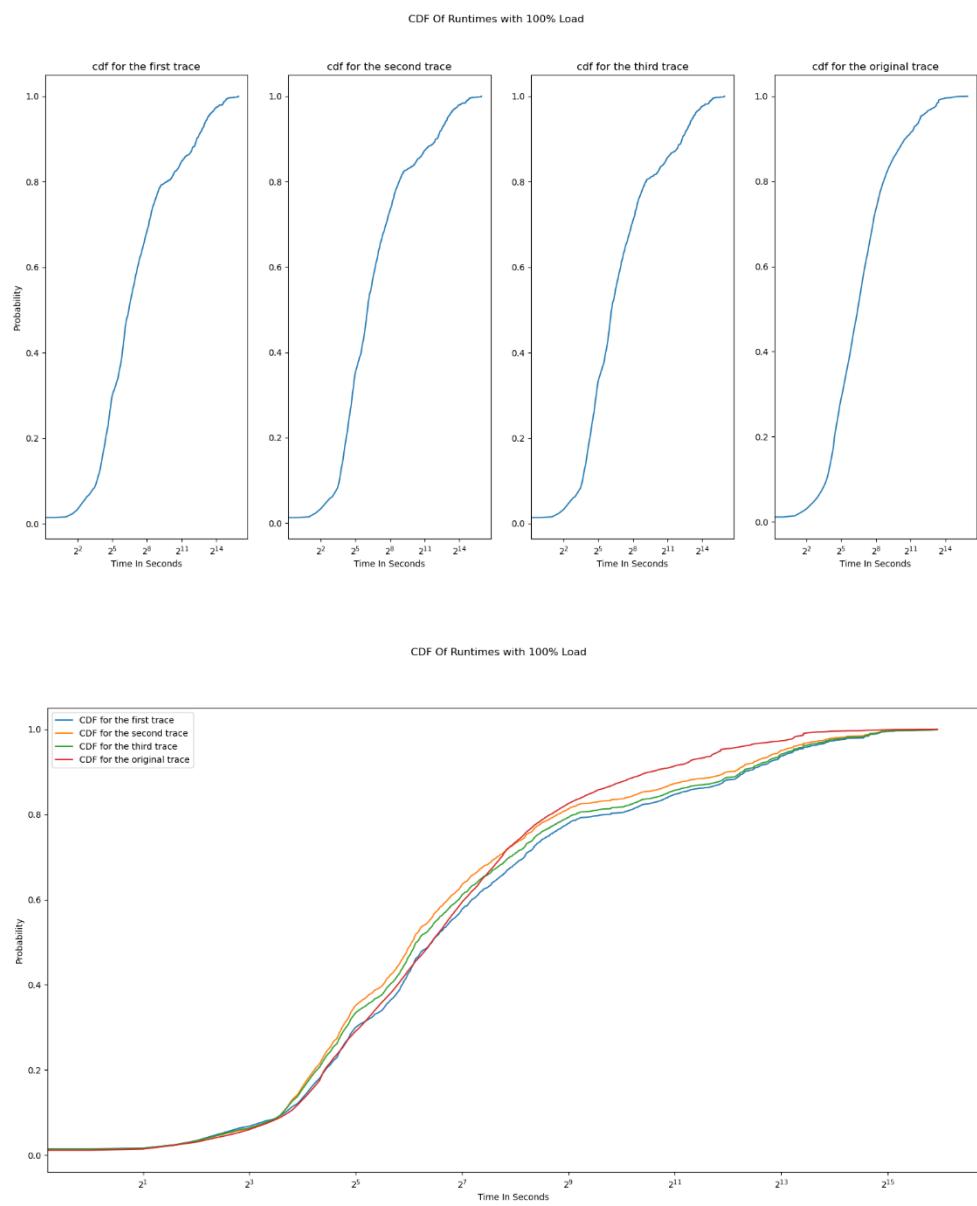


Figure 9-8: CDF of runtimes with 100%

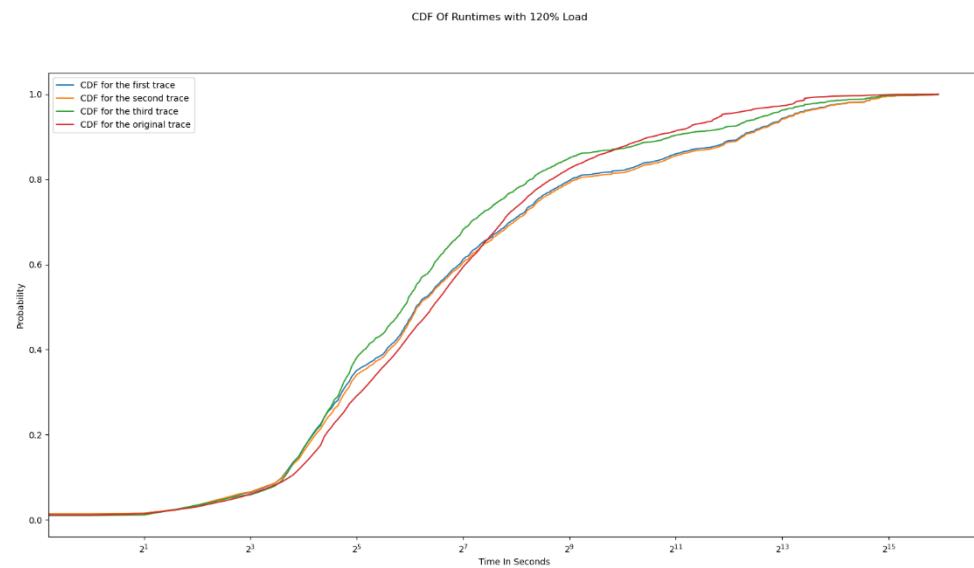
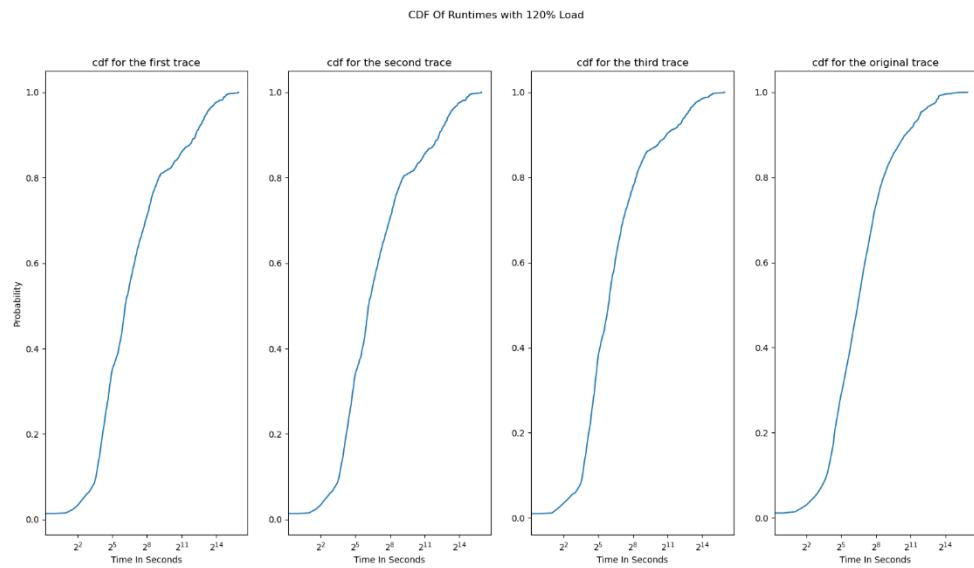


Figure 9-9: CDF of runtimes with 120%

בהתבסבות בליית על האיזרים 7-9 עד 9-9, ניתן לראות שישנו דפוס התפלגות דומה בין שלושת ה-Traces לעומת ה-Trace- המקורי.

:User Distribution .3

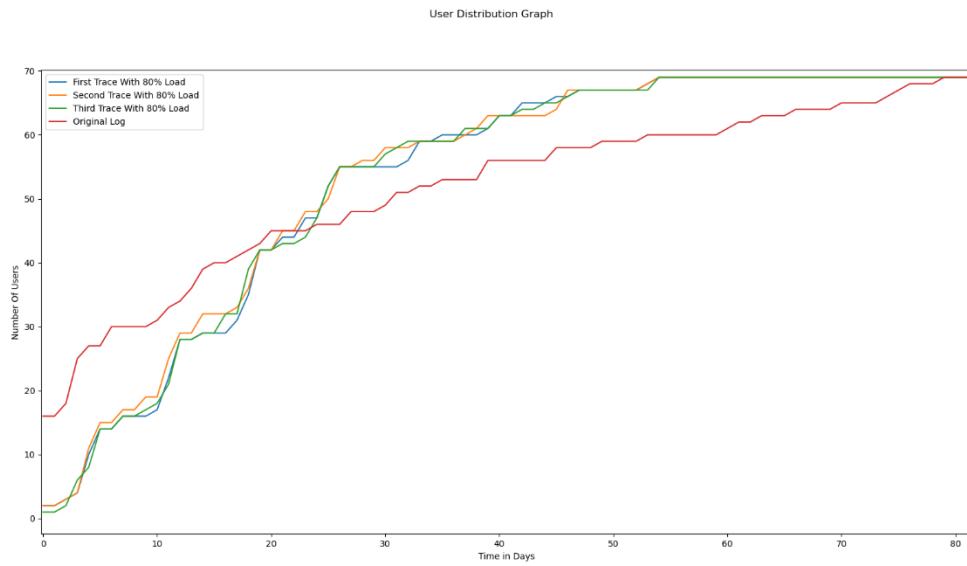


Figure 9-10: 80% Load user distribution

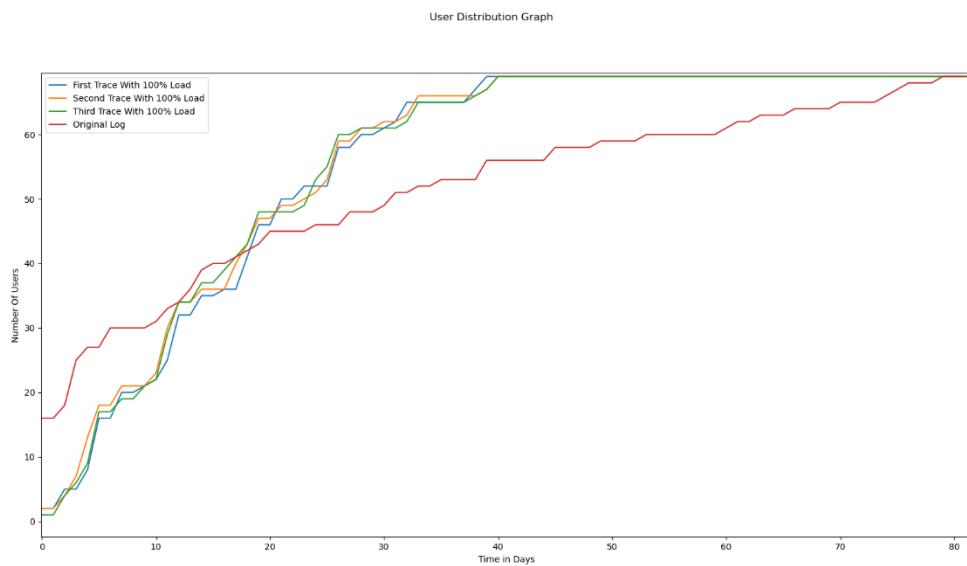


Figure 9-11: 100% load user distribution

בשלושת האיורים (9-10 ו-12-9) ניתן לראות גרפים, עליהם בהतפלגות של-users, באופן דומה ל-Trace המוקורי שלנו. ההבדל בציר האופקי הוא בגל הסיבה שחתכנו את הגראפים, מכיוון שהקו ממשיר באופן יציב עד סוף הימים. ציר האנכי מייצג את מספר-users שראינו עד לרגע זה.

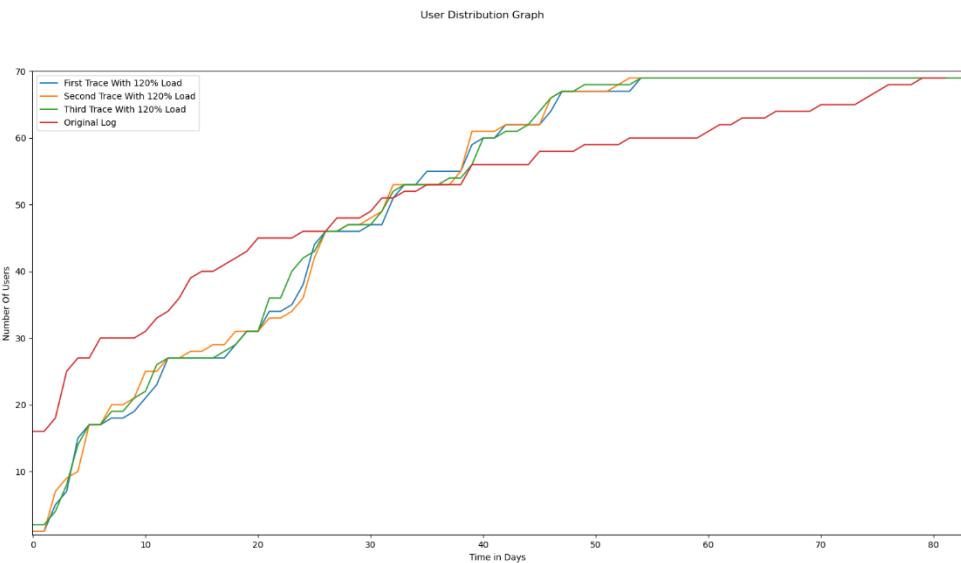


Figure 9-12: 120% load user distribution

3. קורלציה

כדי לבדוק את הקורלציה, בחנו את ה-Job sizes ואות ה-Runtimes Run times לשלוות העומסים ולשלוות ה-Traces, ובנוסף הרינו את ה-Think times ואות זה-times הגרפים מתוארים בהמשך:

:Job Sizes and Run Times .1

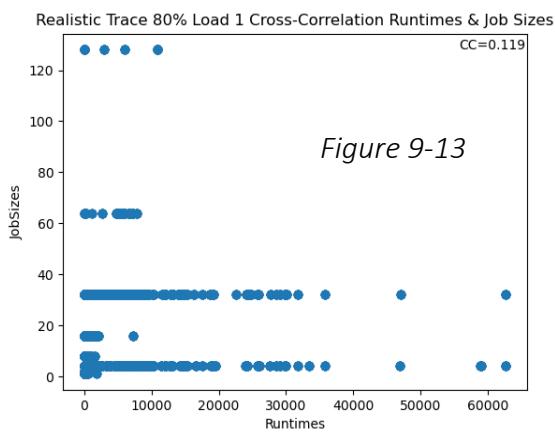


Figure 9-13

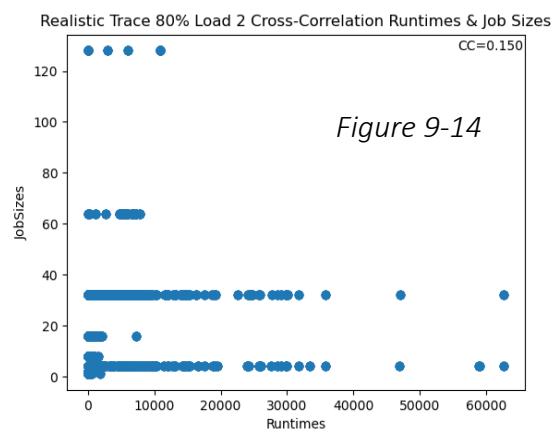
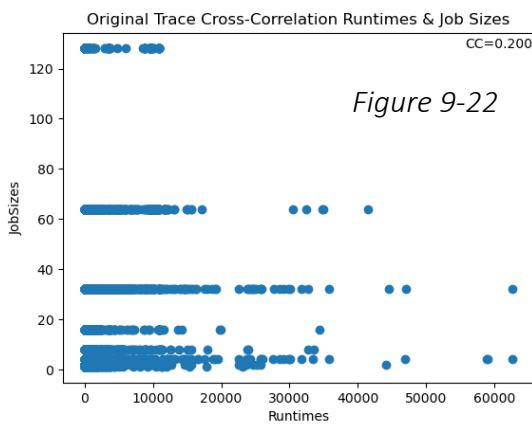
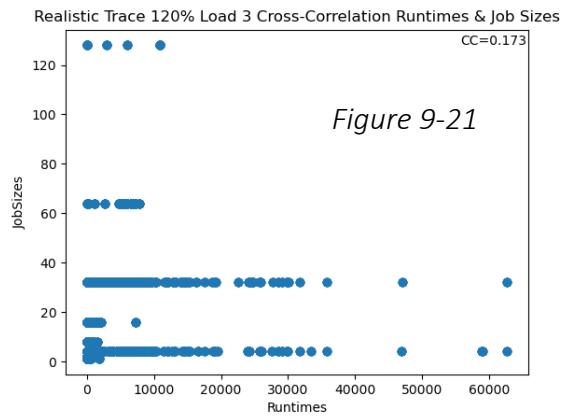
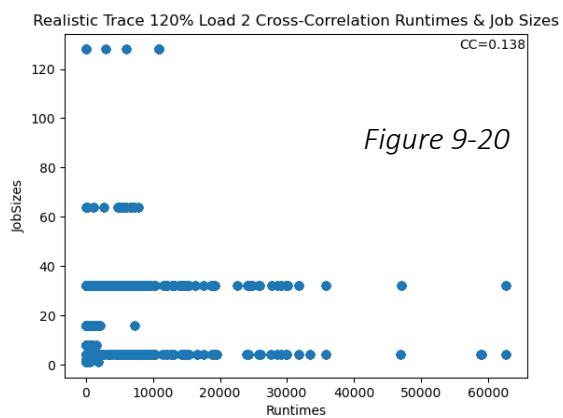
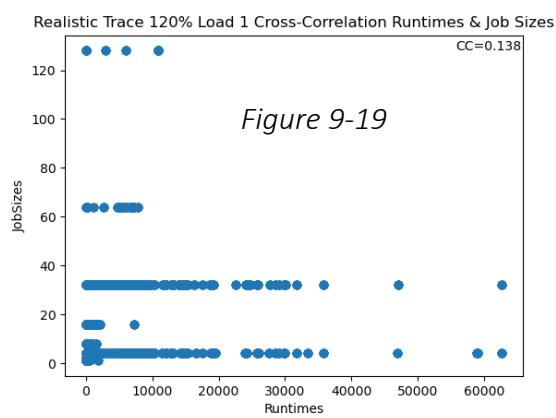
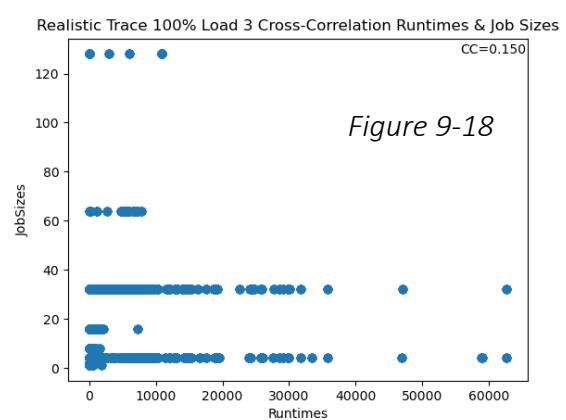
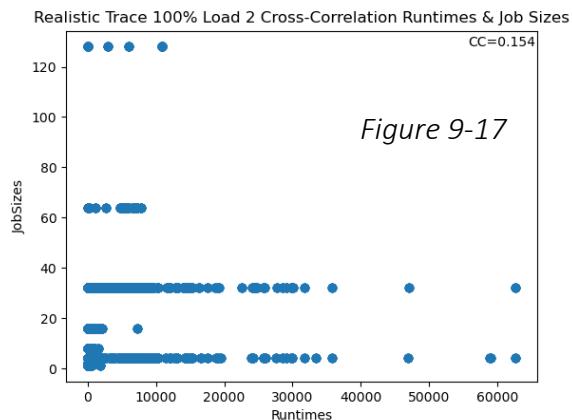
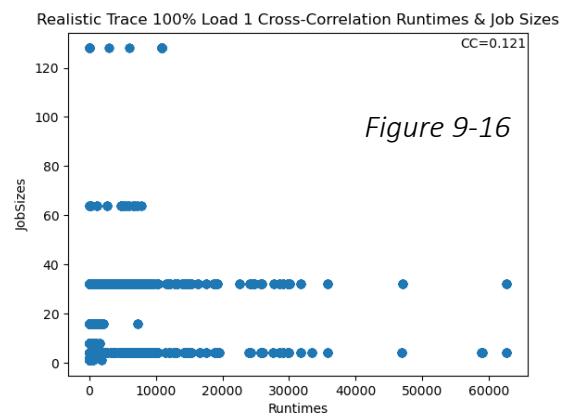
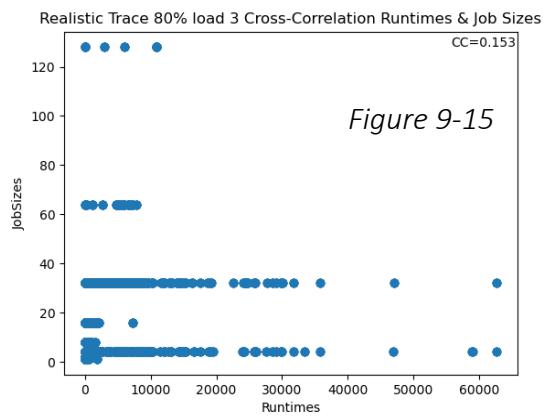
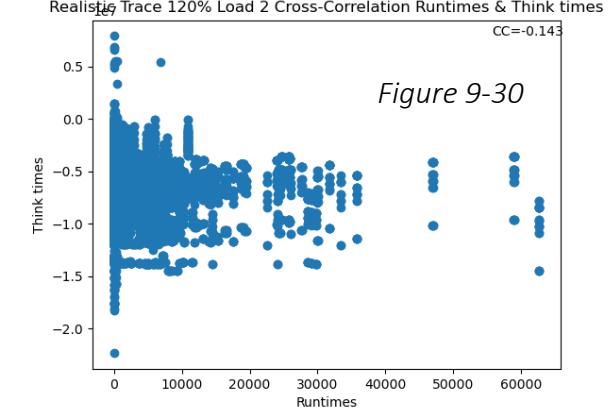
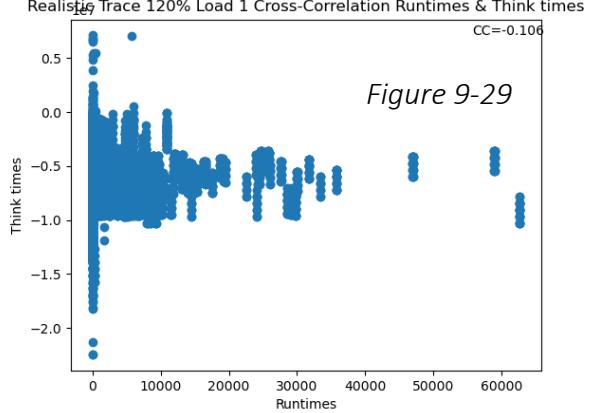
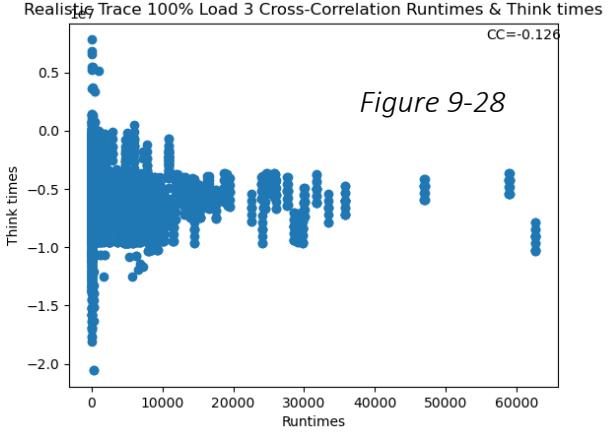
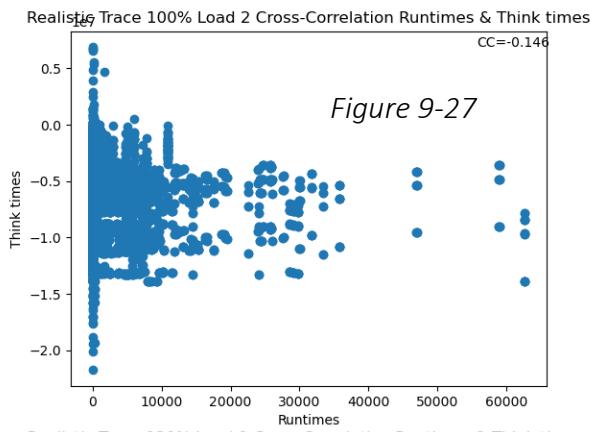
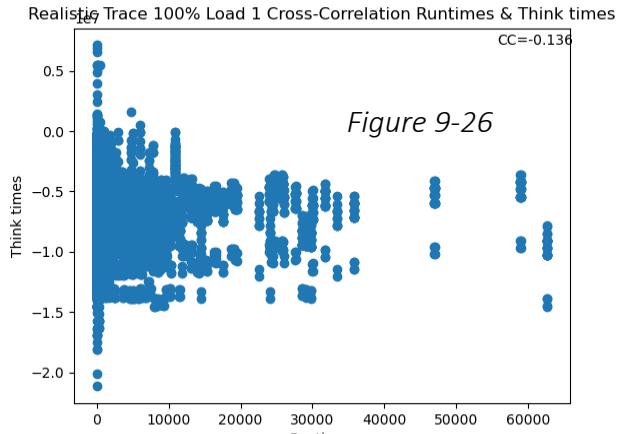
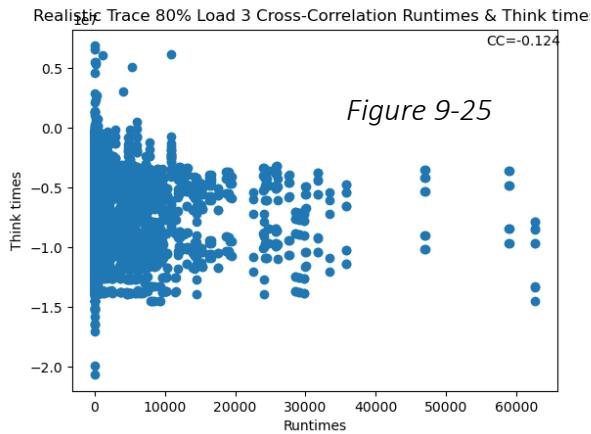
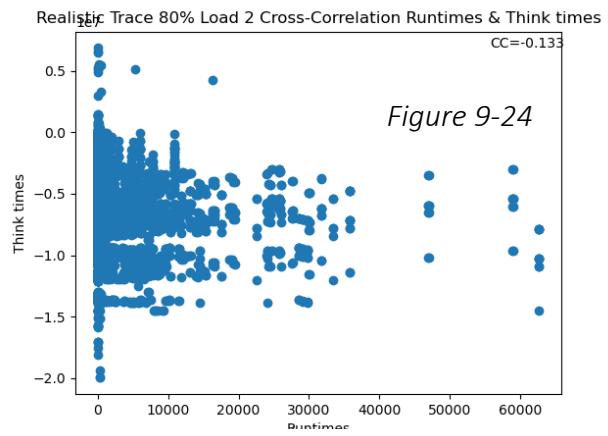
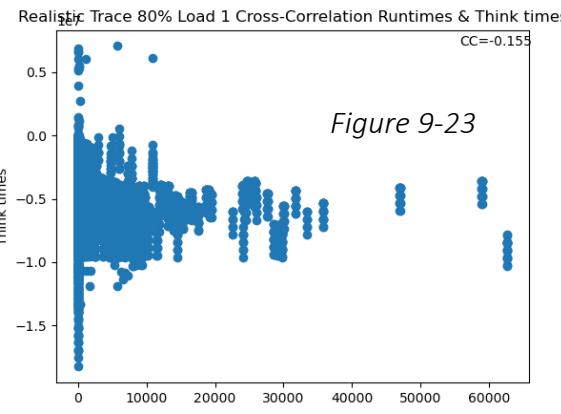


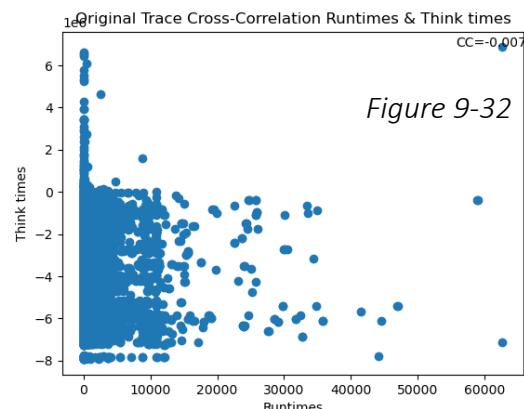
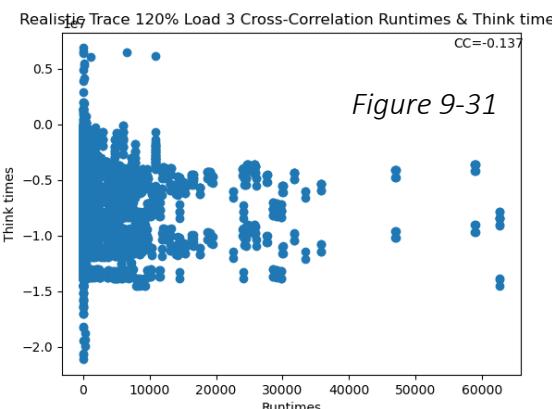
Figure 9-14

ניתן לראות כי מקדם הקורלציה אכן מתפלג בין 0.1 ו- 0.2 בכל הגרפים. כמו כן, ניתן לראות כי לא הייתה לנו קורלציה די ברורה, שכן אין זה המצב בשלושת-Traces שהפקנו ובכל שלושת העומסים.



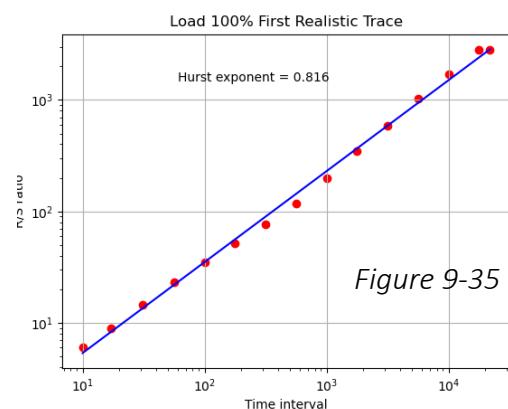
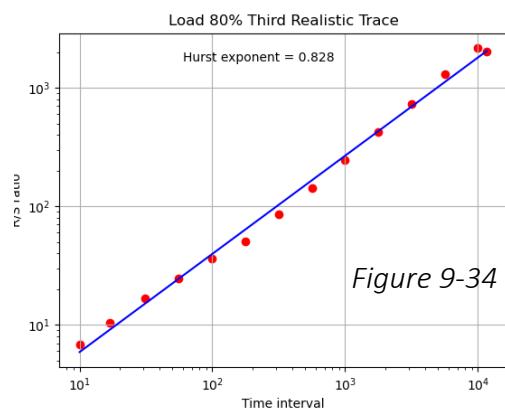
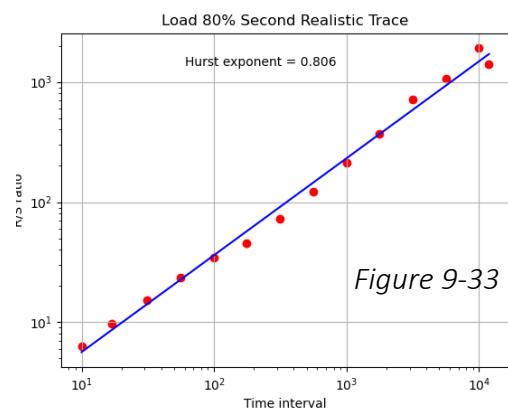
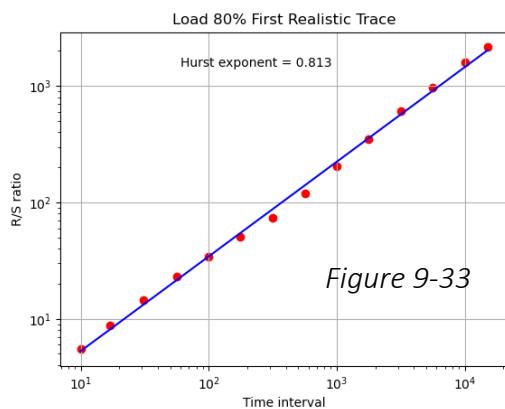
Think Times and Run Times .2

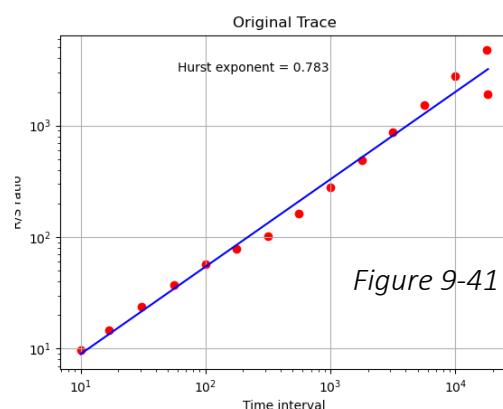
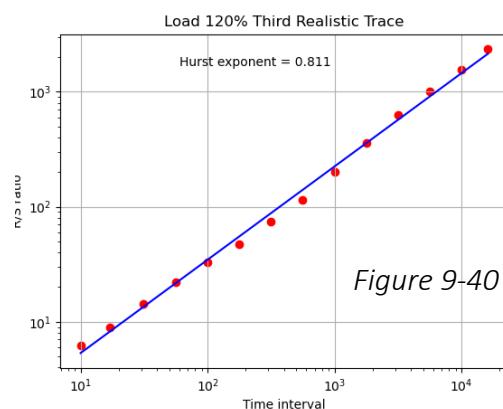
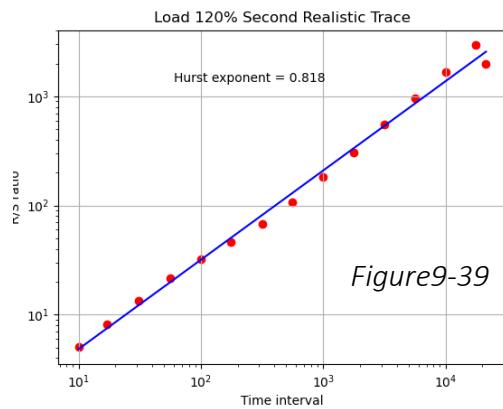
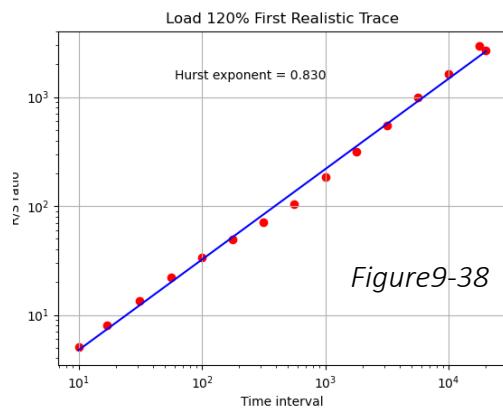
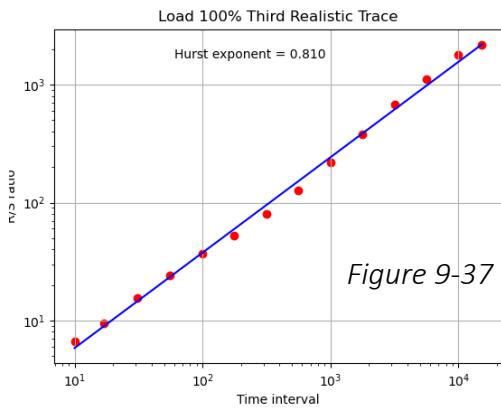
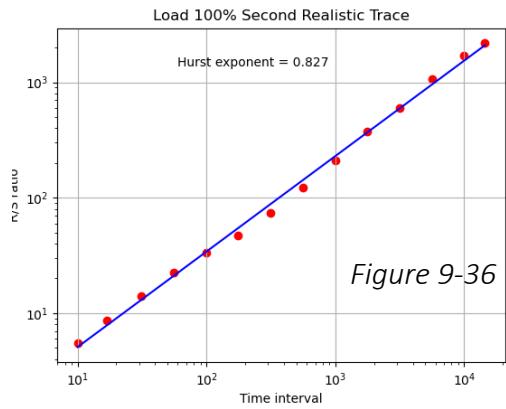




באיורים 9-23 ו-9-32, ניתן לראות כי מקדמי הקורלציה התרפגו בין 0.007- ל- 0.15- בכל הגרפים, גם לפי ההגדרה אין לנו קורלציה, ותמונה זו נשמרת בשלושת ה-Traces שלנו.

Self-Similarity .4





ניתן לראות כי בשלושת ה-Traces, בכל רמות העומס, שמרנו על גרף די דומה, עם טווח הדוק של ה-Hurst

A value of $H = 1/2$ indicates that the process is not self-“

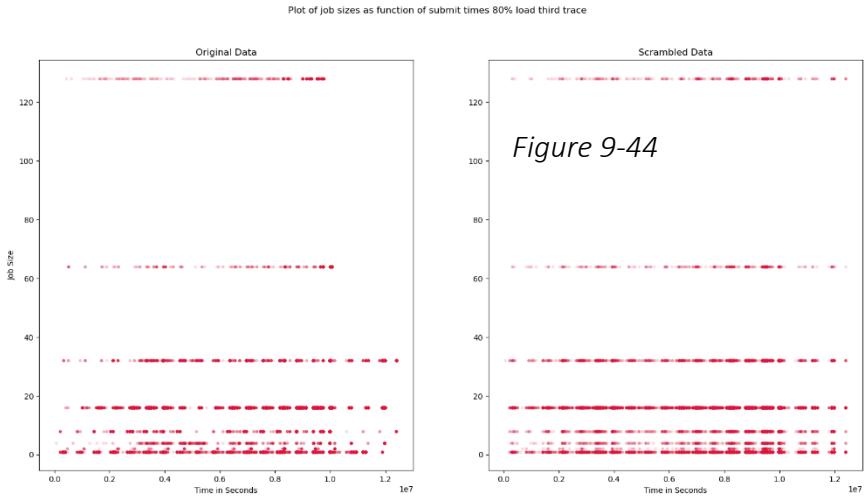
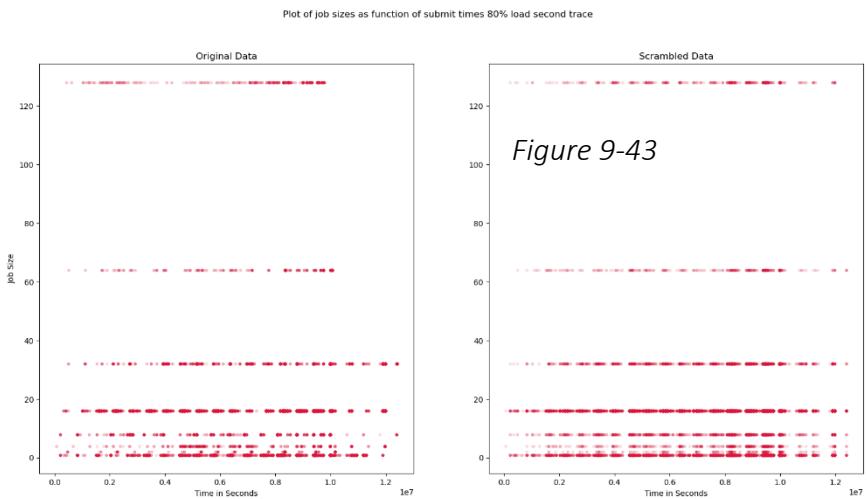
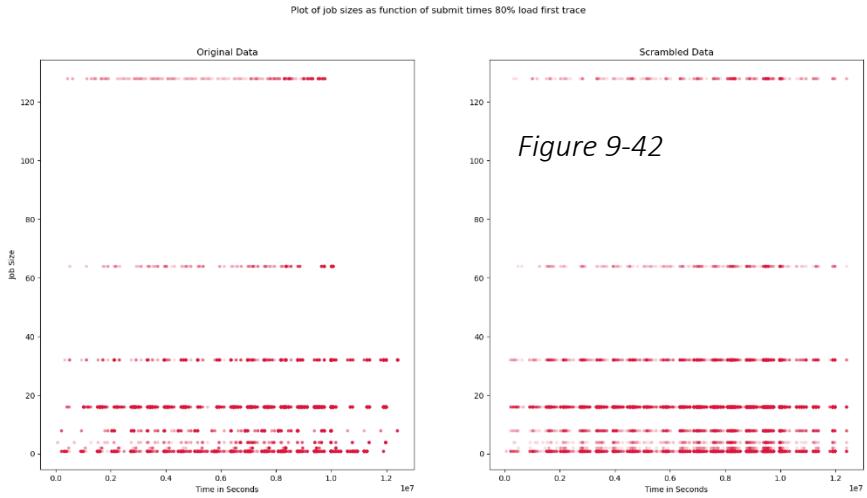
Exponent similar, but conforms with a Poisson process. A value of H in the range $1/2 < H < 1$ is taken to

² mean that the process is indeed self-similar

הגרפים פה מייצגים בסקללה לוגריתמית את ה Rescaled Range בתלות בזמן.

Locality of Sampling .5

בגרפים מוצגים Job Sizes כפונקציה של Submit Times



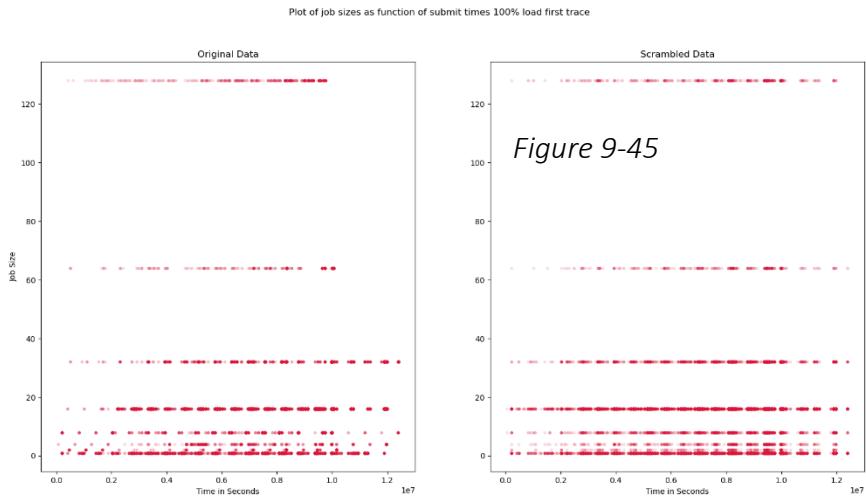


Figure 9-45

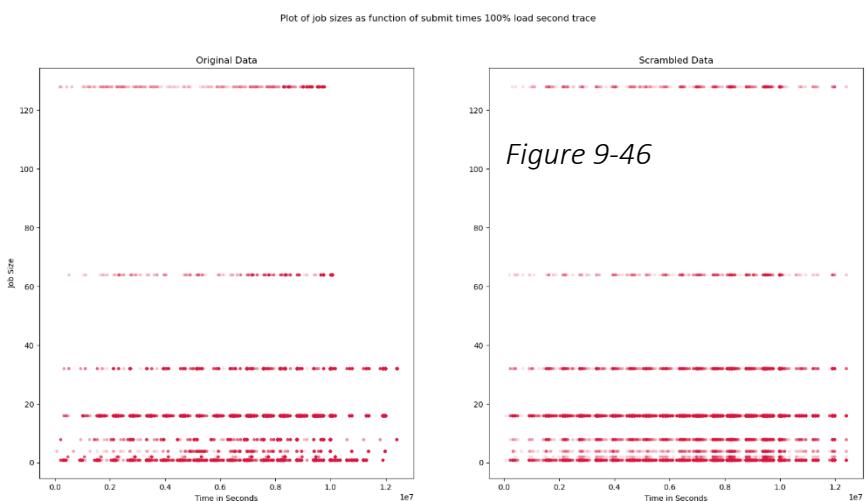


Figure 9-46

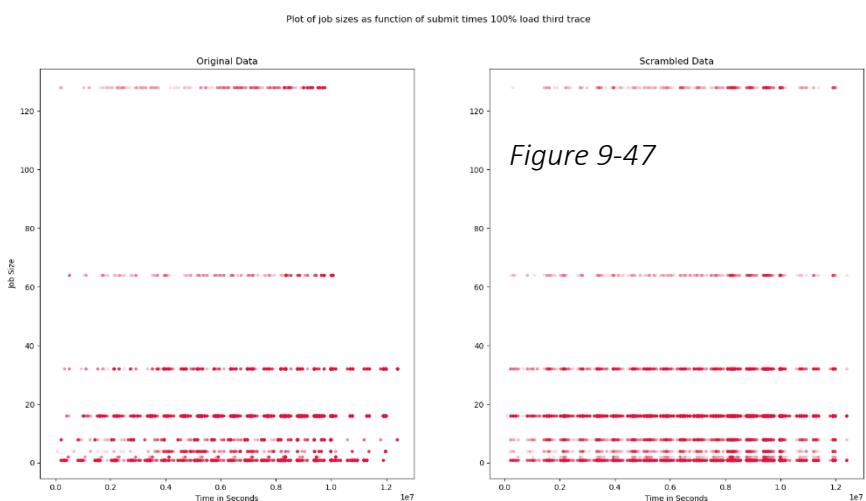


Figure 9-47

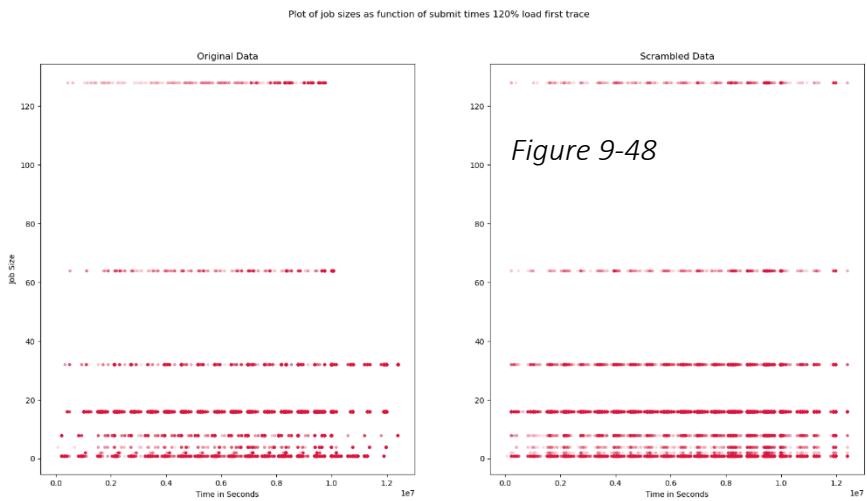


Figure 9-48

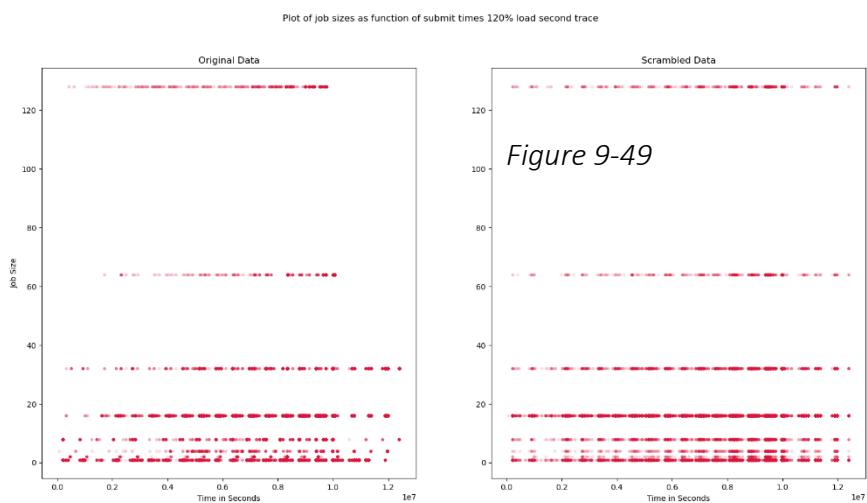


Figure 9-49

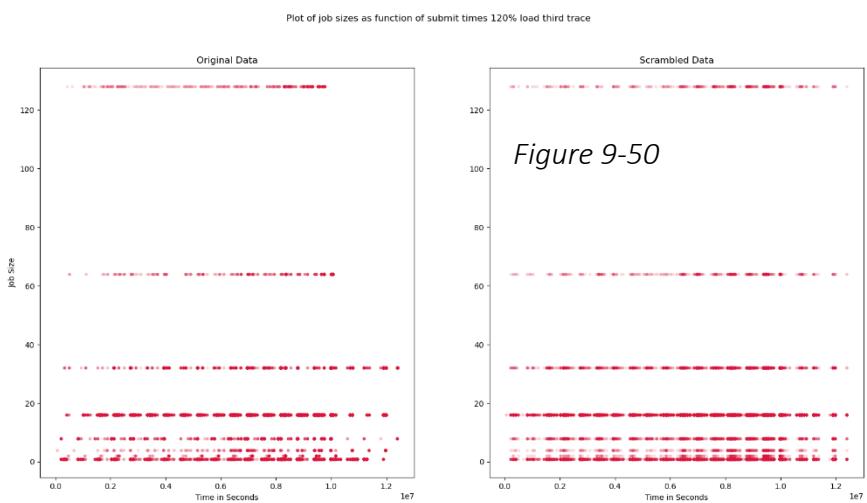


Figure 9-50

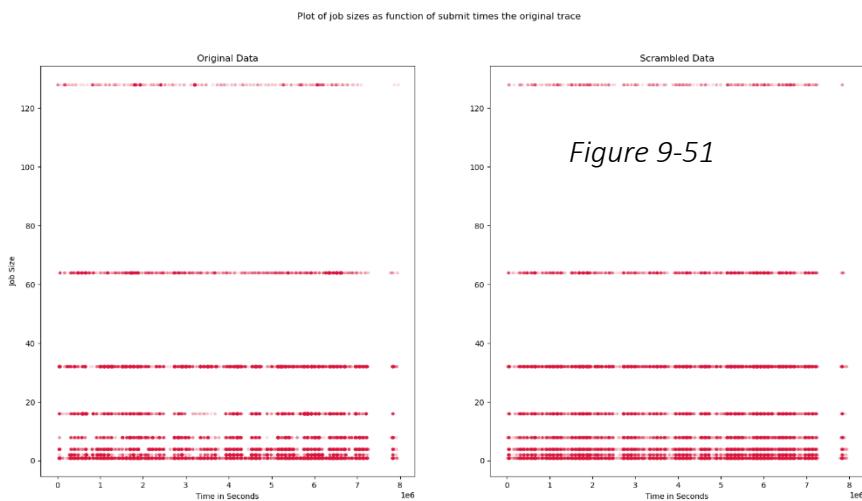


Figure 9-51

אם נסתכל על איור 9-42 ועד 9-51, נראה כי ה-Locality of Sampling נשמר בכמה וכמה נקודות למרות הירוב שהציגנו, ובנוסף, גם אם נסתכל על כל גרסה של כל Trace בנפרד, במיוחד באזורי אשר בו ה-Job Sizes שוים ל-20, 40, 60 ו-120, ניתן לראות כי קיימים דיזאינר זומה בין כל הגרסאות, עם נקודות בהירות וכיהות, מה שמעיד לנו על ה-Locality of Sampling כפי שהוזג בספר³.

Submission Rate .6

בסעיף זה, מתוארכות ההשוואות בין ה-Submission Rates של Users, בהתאם לעומס שהמערכת נמצאת בו. גרפים 9-52 עד 9-52 מתראים לנו بصورة גרפית (ECDF) לכל עומס במערכת ובכל Trace.

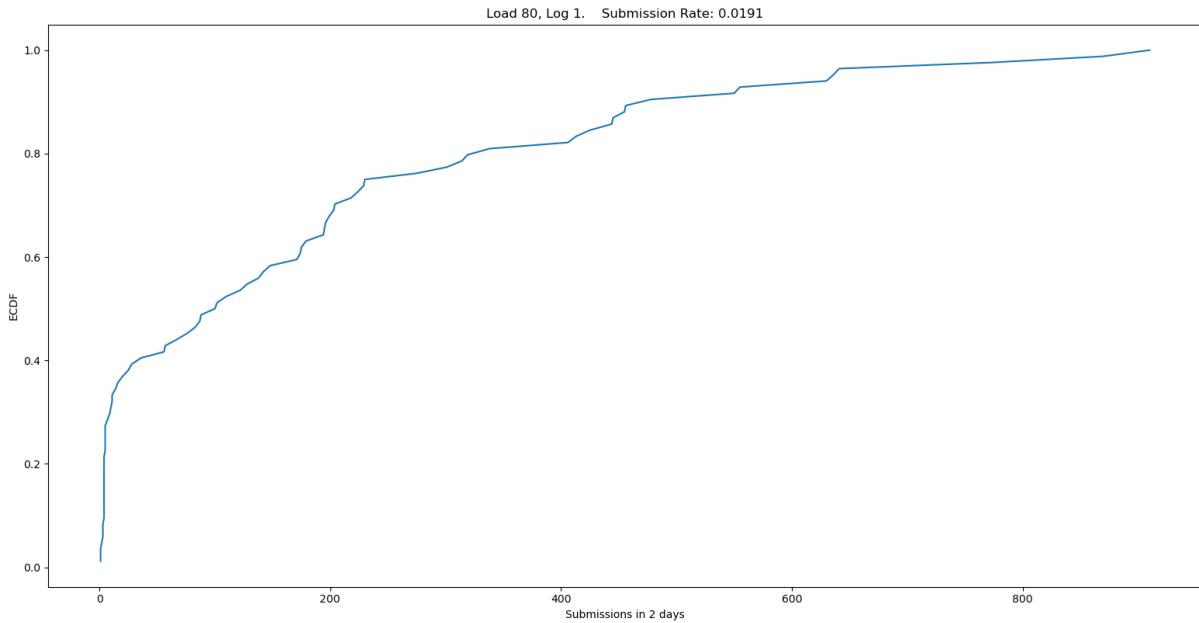


Figure 9-52: Load 80%, Trace 1

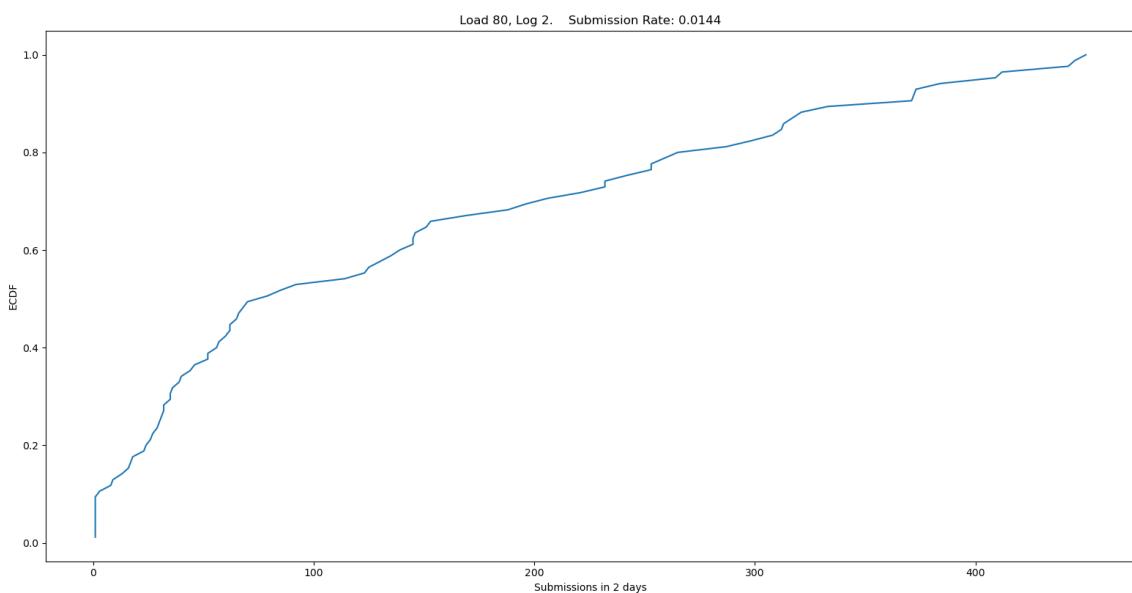


Figure 9-53: Load 80%, Trace 2

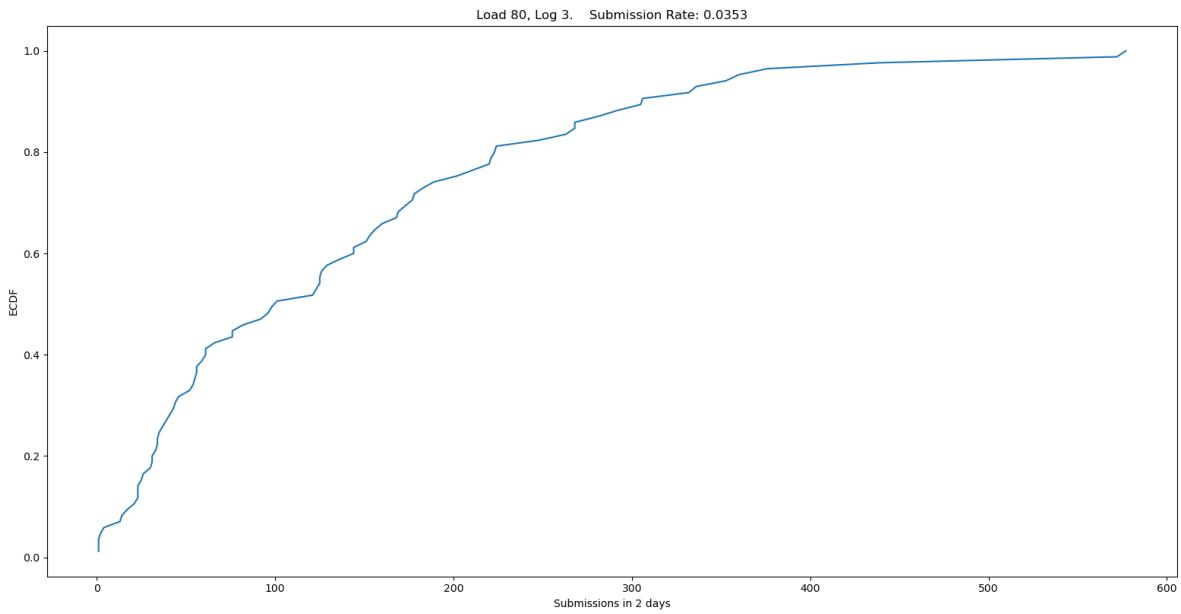


Figure 9-54: Load 80%, Trace 3

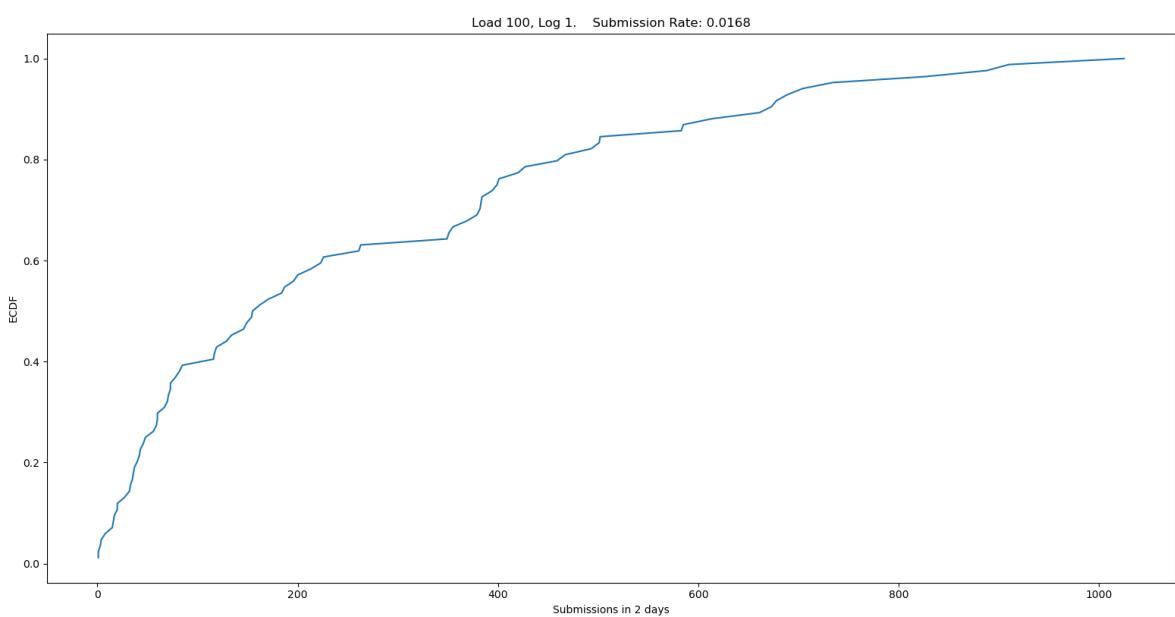


Figure 9-55: Load 100%, Trace 1

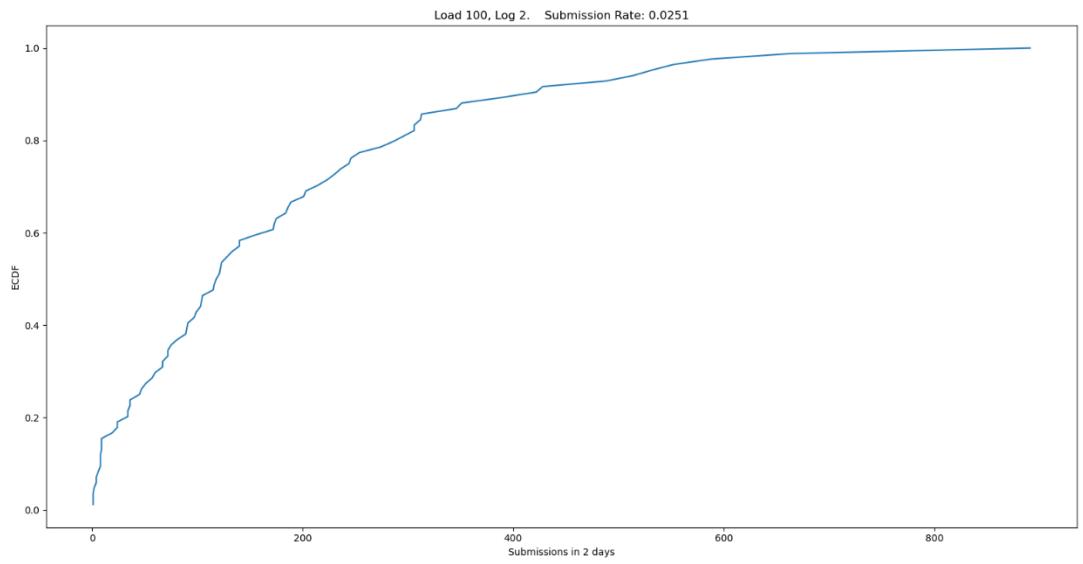


Figure 9-56: Load 100%, Trace 2

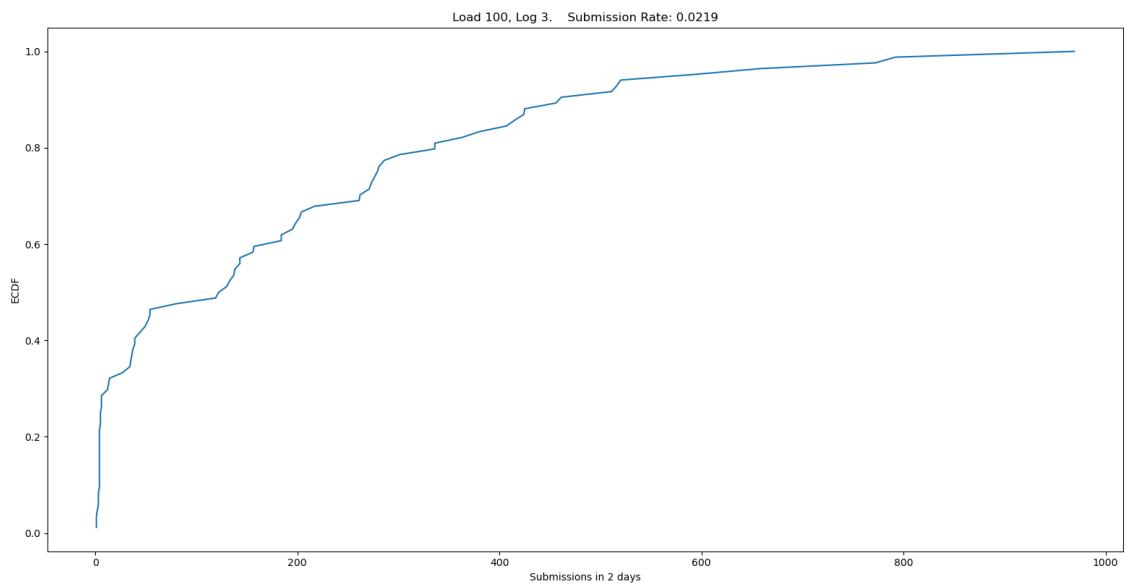


Figure 9-57: Load 100%, Trace 3

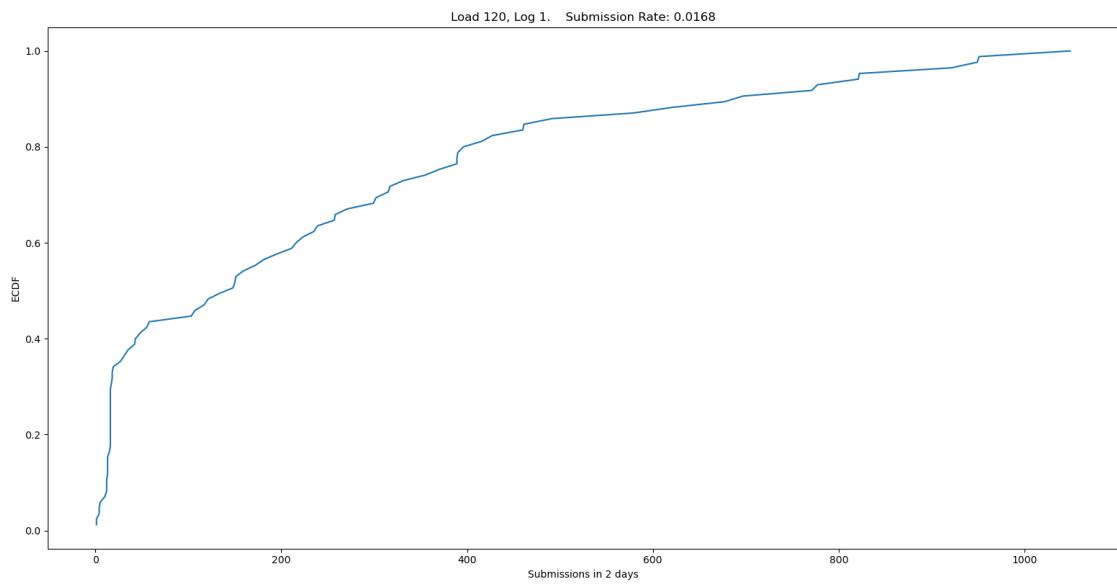


Figure 9-58: Load 120%, Trace 1

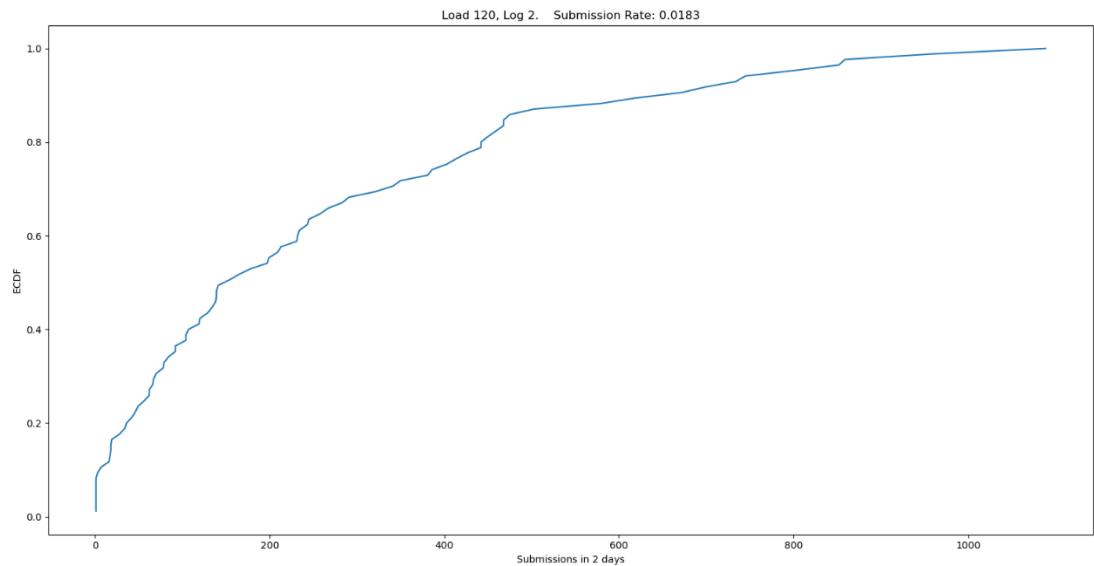


Figure 9-59: Load 120%, Trace 2

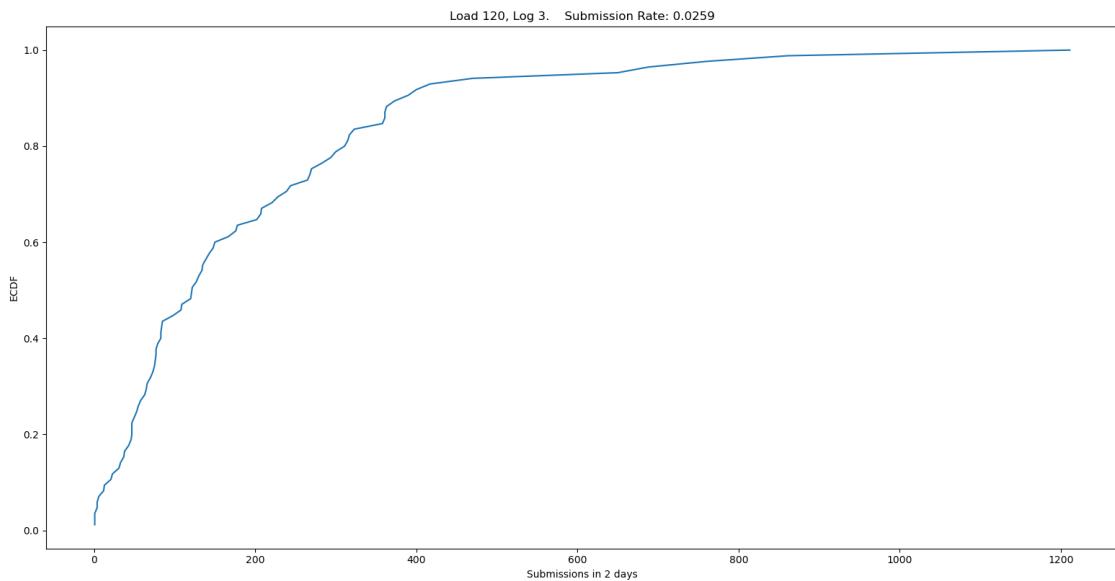


Figure 9-60: Load 120%, Trace 3

Submission Rate=average (1/Interarrivals)

לפי השיחה שהייתה על ה Submission Rate סיכמנו שה Submission Rate הוא ההופכי של Interarrivals لكن חישבנו את ה Submission Rate לפי מה שנקבע והציגנו גם גרפית את מספר ה submissions תוך פרקי זמן של יומיים ביחס ל ECDF.

ה Submission Rate לא נותן לנו הבדלה בין העומסים וזאת יכול להיות מהסיבה שהמשתמשים שנבחרו ב-traces עם פחות עומס jobs שלהם צפויים יותר.

אבל אם נסתכל על גרפי ECDF אפשר לראות שעם עומס של 80% 80% מספר הגашות בחלון של יומיים היה עם מקסימום של 900-600, עם עומס 100% היה לנו מקסימום של 1050-900 ועם עומס 120% מקסימום של 1250-1150 בקירוב.

אם נעשה ניתוח עמוק יותר, נסיק כי הערך של submissions תוך חלון של יומיים עולה מעל 400 עם ערבי ECDF של 0.8-0.93 ב-traces 0.78-0.8 עם עומס של 80%, וערבי ECDF של 0.7-0.8 ב-traces 0.7-0.8 עם עומס של 100% וערבי ECDF של 0.7-0.8 ב-traces 0.7-0.8 עם עומס של 120%. מה שבעצם הניתוח הזה אומר, ככל שעולים בעומס, יש לנו אחוז גדול יותר של חלונות בגודל יומיים עם 400 או יותר הגашות.

Trends and Cycles .7

נתחיל עם השוואת הטרנדים בגרפים.

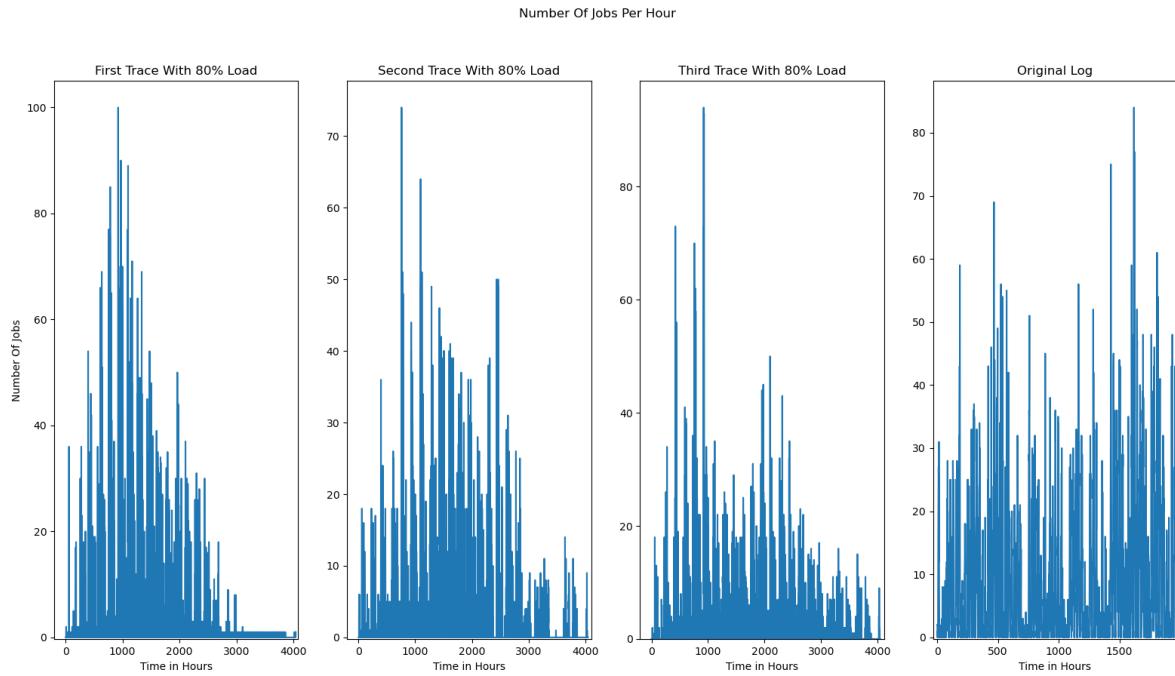


Figure 9-61

ראשית נסתכל על ה-Traces עם 80% load (תמונה 61-9). ניתן לראות שאנו בודקים את מספר ה-jobs שמתבצעים בשעה. ניתן לשים לב שישנו דמיון כל בין ה-Traces שייצרנו לבין ה-Trace המקורי. בחלק משיטת ה-User resampling שבחרנו, יש לנו מספר Users שונים בחורם בכל שבוע מחדש באופן רנדומלי. בעקבות סיבה זו, קשה לתרנד להישאר נאמן לאופי הטרנד המקורי.

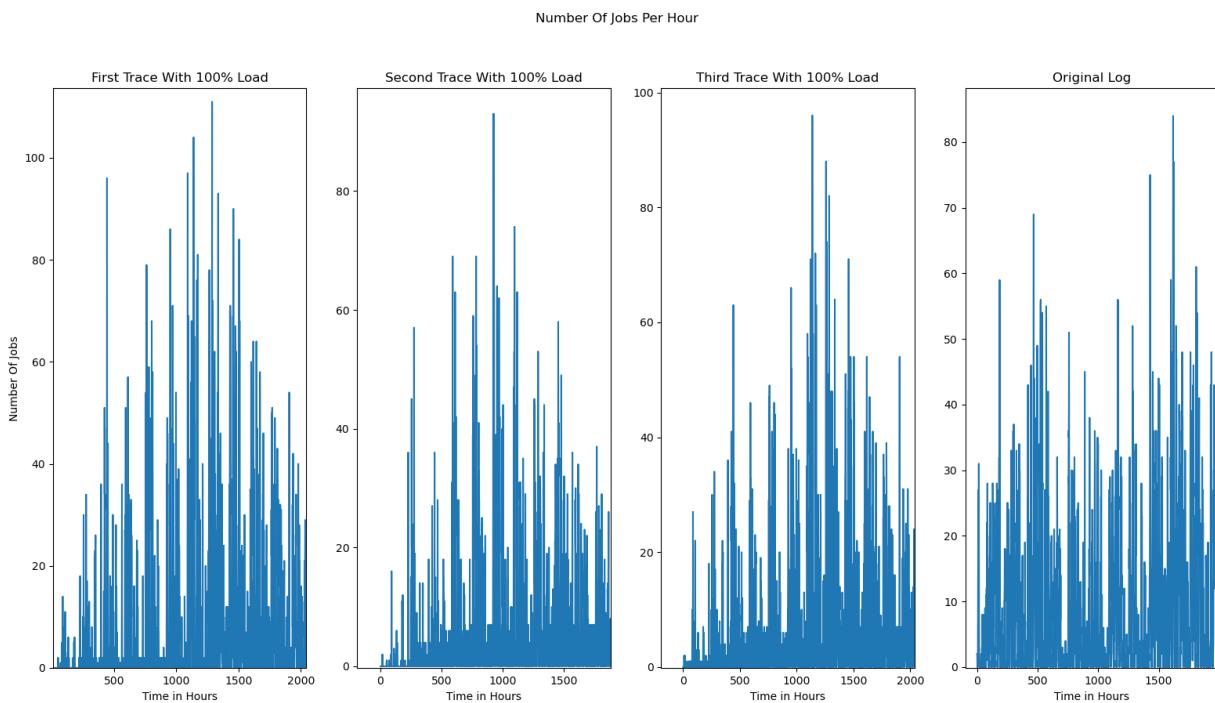


Figure 9-62

במהר לך, ב-*load* 100% עם Traces (תמונה 9-62), נשים לב שינוי שוני כל בין התוצאות, כאשר ב-*Trace* המקורי ישנים שני "פוקים" שונים לב אליהם, בעוד ב-*Traces* שלנו יש "פוק" אחד יותר דומיננטי, זאת בעקבות שיטת בחירת *Users*, אשר "שוברת" לנו את הטרנד.

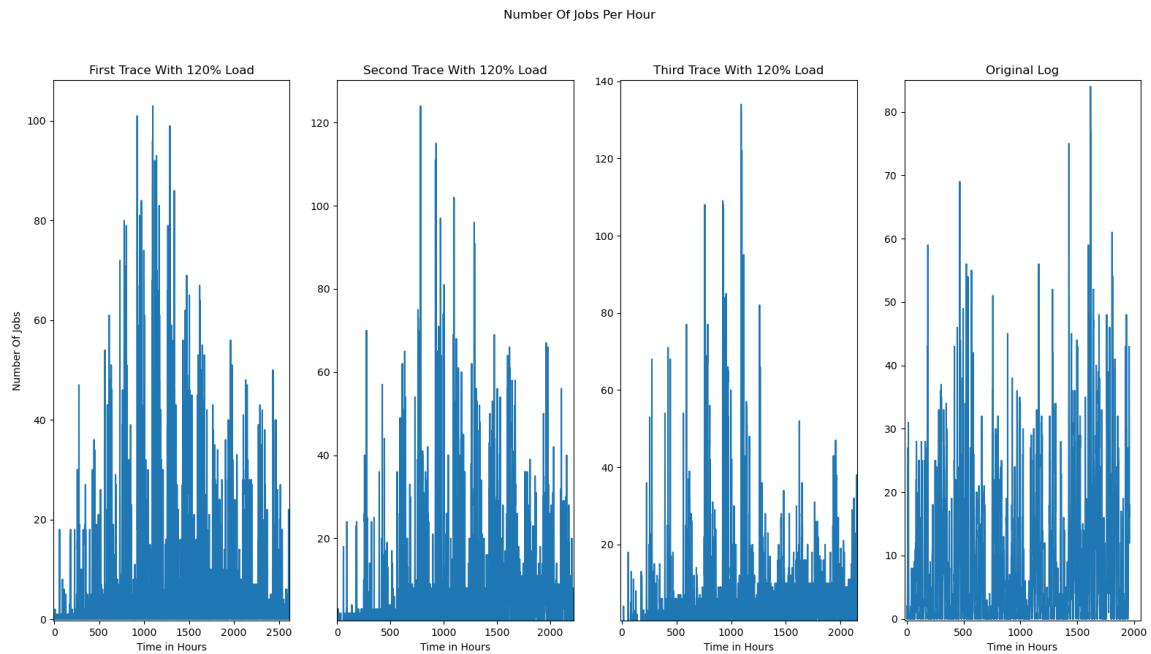


Figure 9-63

וב-*load* עם 120% Traces (תמונה 9-63), בדומה לתיאור של ה-*Traces* ב-80% וב-100%, נראה כי קיים דמיון כל (מצהים שני "גלים" בהיסטוגרמות, אך לא כמו ב-*Trace* המקורי).

בעת נאבחן את ניתוח *Cycles*.

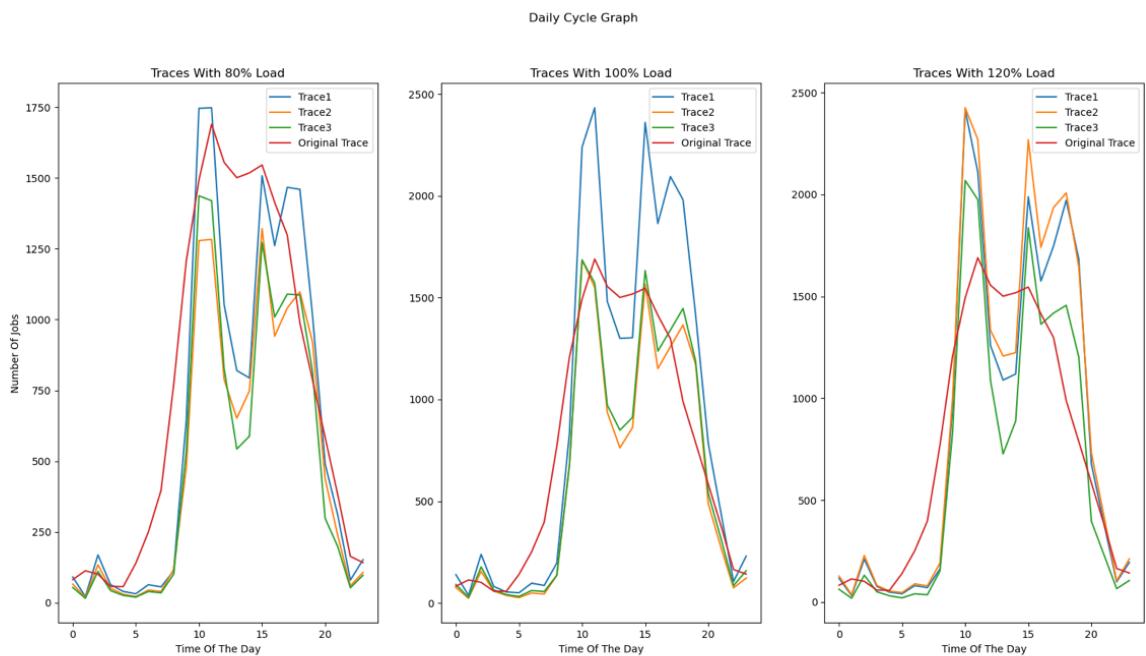


Figure 9-64

ראשית נביט על תמונה 9-64 אשר מתארת לנו מחזוריות יומיות. בכל אחד מהגרפים, מהתאים שלושת ה-Traces שלנו ובנוסף גם המלורי. נשים לב, כי למעט זמנים ספציפיים (בין השעות 11-14) אשר בהם המגמה של הגרף דומה אף הרכים לא דומים, כל שאר שעות היום מואוד דומים. ניתן בקלהות להוות את שעותימי העבודה, את הפסקות (השעות הממוצעות של הפסקה), את סיום העבודה ואת הלילות. נראה כי למעט Trace אחד ב-100% load, כל ה-Traces אחרים דומים גם מבחינת הרכים וגם מבחינת המגמה המחזורית ל-Trace המקורי.

בעת נביט בגרף של מחזוריות שבועית (תמונה 9-65), המתואר באותו אופן של הגרף הקודם. זהה כי גם בגרף שבועי, הדמיון נשאר מואוד דומה ונitin בקלהות להוות מחזוריות של שבוע – ימי חול ימי חופש. נראה כי בחלק מה-Traces, הרכים אמורים שונים מה-Trace המקורי, אך המחזוריות נשמרות בצורה טוביה מאוד למראות מגבלות שיטת יצירת ה-Traces.

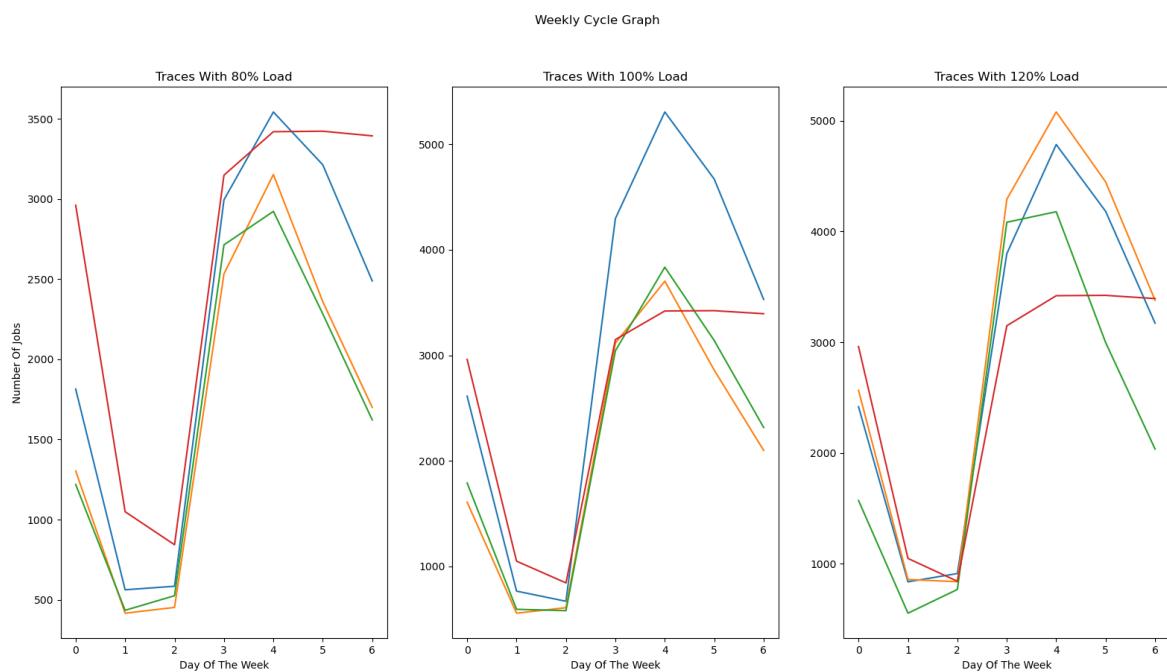


Figure 9-65

Wait Times . 8

ראשית נראה את גרף ה-ECDF של wait times תחת loads שונים (תמונה 9-66).

נשים לב כי בכל trace,wait times אך ההסתברות לקבל אותם משתנה. זה קורה מכיוון שאנו דוגמים את אותם users בטעמים שונים – מה שמשנה את ההסתברות לכל time.

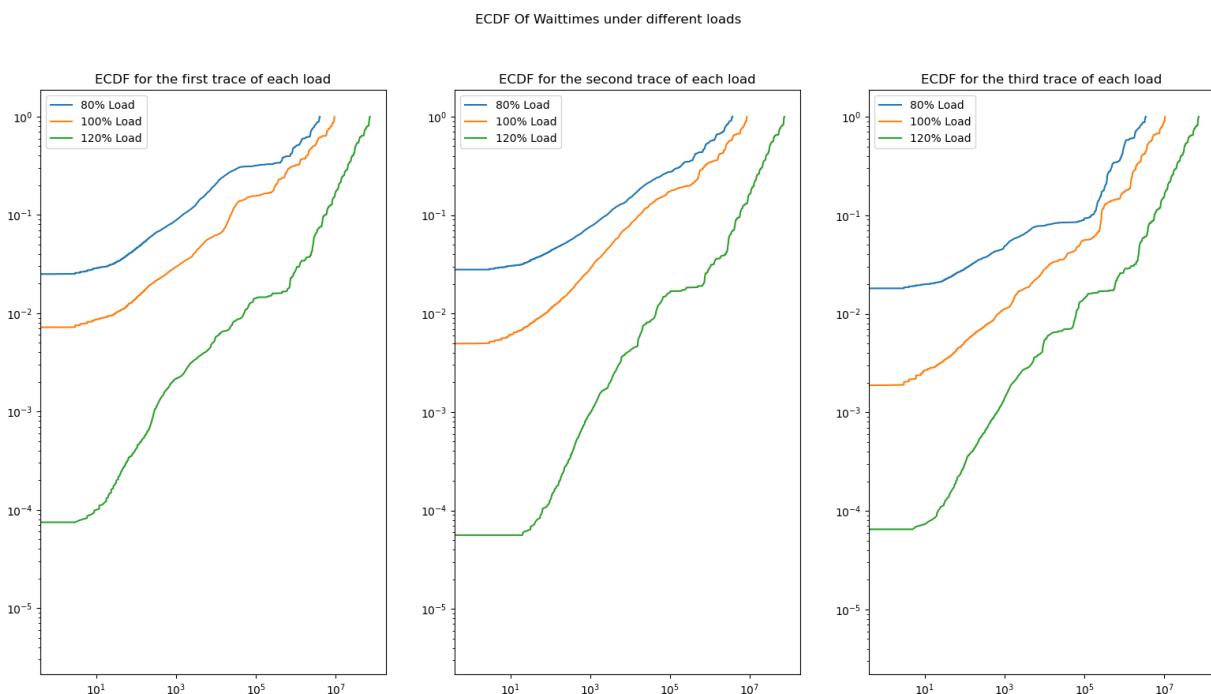


Figure 9-66

באיורים של scatterplot ראינו שיש תבנית מסוימת בכל הגрафים ולכן חשבנו על זה שיש קורלציה בין wait times ו submit times , ואכן מתרור שמקדם הקורלציה הולך ומתקרב ל 1 בכל שהעומס עולה. ההסבר לה הוא שיש לנו 9 משתמשים שאנו דוגמים מהם, אז ה waittimes לגיבים של אותו משתמש הם קבועים ולפי השיטה שלנו אנחנו דוגמים שוב ושוב מאותם משתמשים, ז"א אוטם waittimes ואוטם runtimes מוביל אותנו לקורלציה חזקה בין שני המשתנים האלה. אבל גם אפשר לראות הפרש של 0.1 במקדם הקורלציה בין כל רמת עומס לרמה הבאה, וזה לא מפתיע כי כמו שאמרנו בשלב 8 השיטה שלנו להגדלת העומס היא בדגם חזרת של אותו משתמש מה שגורם להוספה עוד אותו זוגות של waittimes ו submittimes מה שמנגדיל את מקדם הקורלציה.

Scatter Plot Of Wait Times As Function Of Subtimes

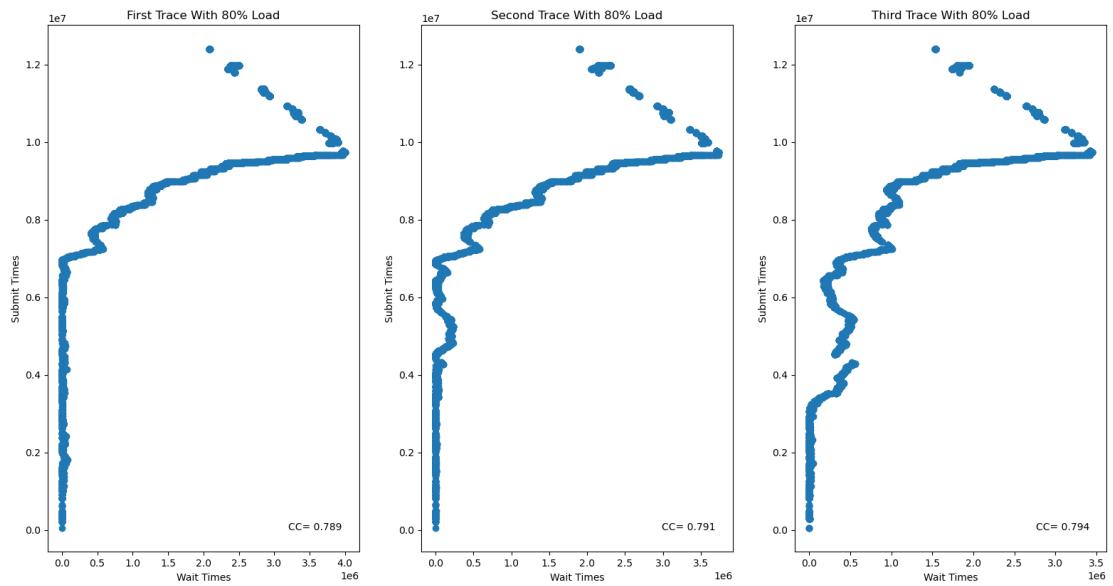


Figure 9-67: Scatterplot of 80% load

Scatter Plot Of Wait Times As Function Of Subtimes

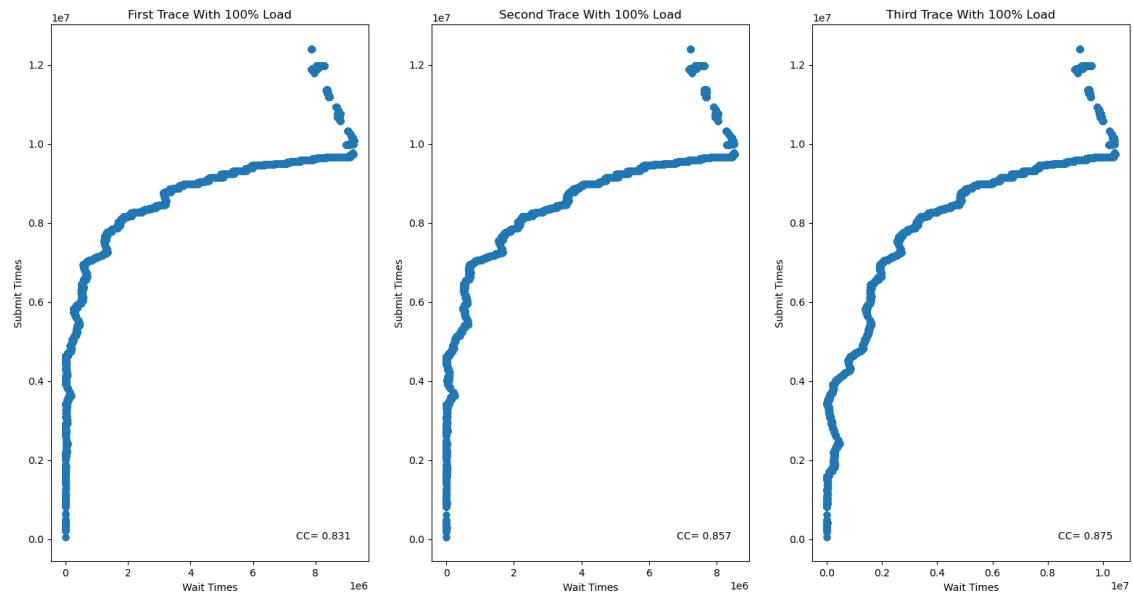


Figure 9-68: Scatterplot of 100% load

Scatter Plot Of Wait Times As Function Of Subtimes

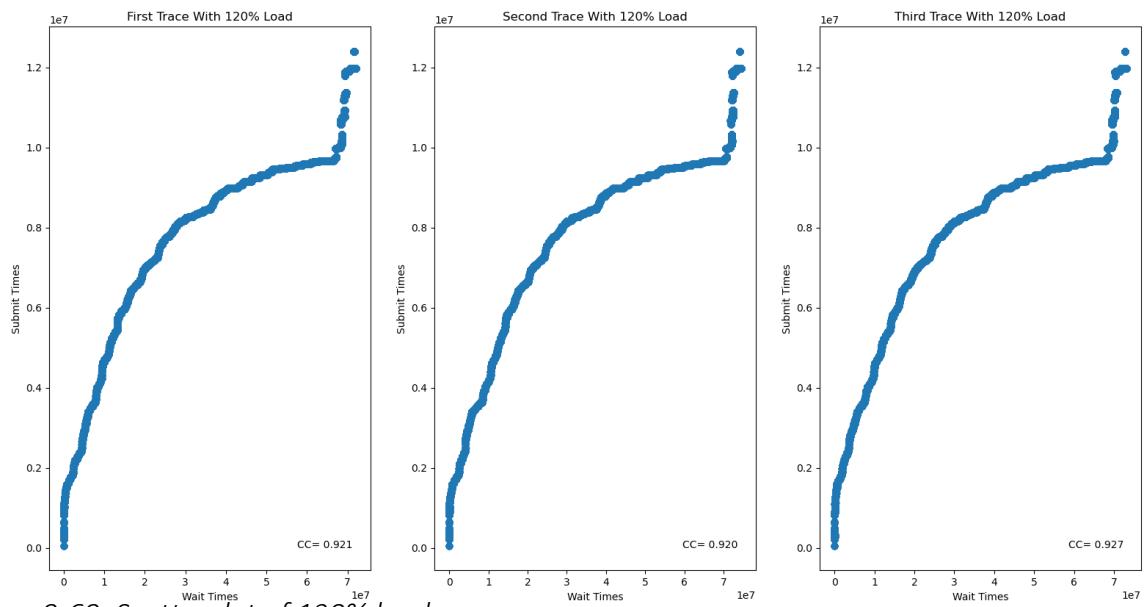


Figure 9-69: Scatterplot of 120% load

Different Analysis – Think Time . 9

בסעיף זה בחרנו לנתח ולהשוות את ה-Traces שהפקנו אחד עם השני ועם ה-Trace המקורי על פי ה-Think times שלהם. נשים לב שהאירועים הבאים מתארים זאת בצורה גרפית. מצד אחד, נראה כי קיים דמיון רב בין ארבעת ה-Traces. אכן ארבעתם מתקדמיים באותה מגמה והגרפים שלהם שומרים על אותה מונוטוניות לאורך כל הדרכן. מצד שני, ערכי ה-Think time מעת שונים זה מזה. ב-Traces שלנו, טווח הערכים גבוה יותר מממוצע – דבר אשר לא מפתיע אותנו כאשר יש עומס רב על המערכת (load 120%) אך קצר גבהה מיידי כאשר המערכת פחותה עומסה. בנוסף, טווח הערכים של ה-Tracethink time ב-Traces שלנו מתחילה ציר ה-X (1.4) לעומת trace המוקורי אשר מתחילה ב-(0.8). ההבדל זה יכול לנובע מגוון סיבות, אך ככל הנראה הבדל זה קורה בעקבות שיטת ה-User Resampling, ובגלל שהוא דוגמים כמות מוגבלת של Users.

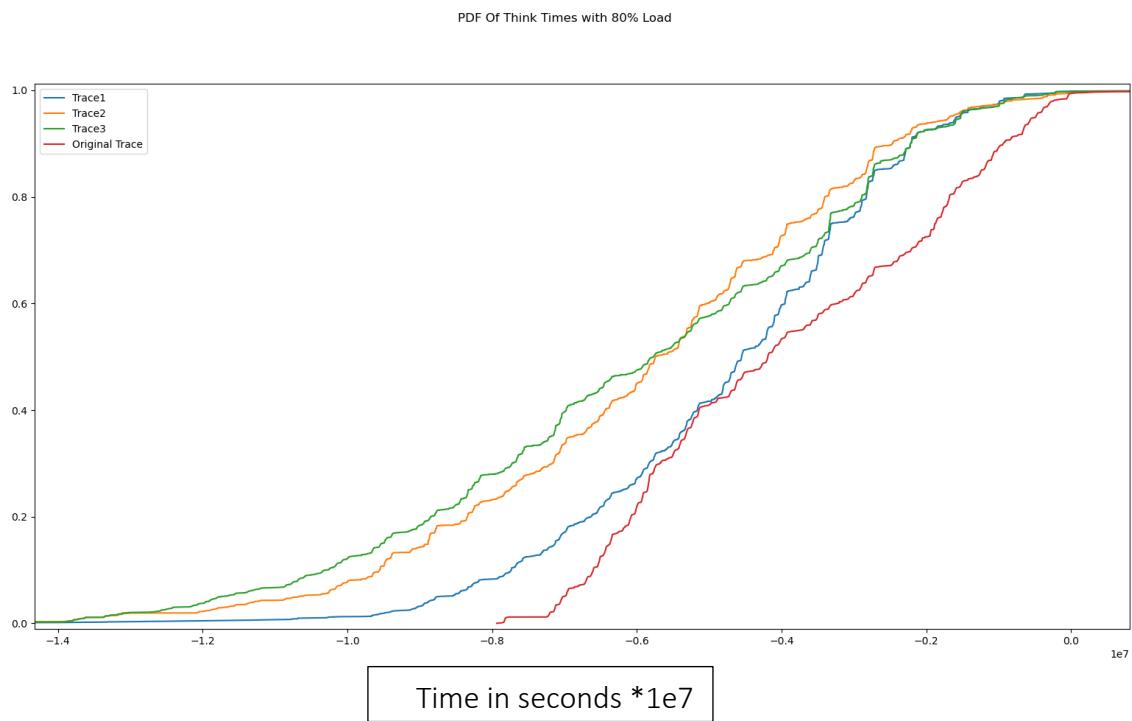


Figure 9-70

PDF Of Think Times with 100% Load

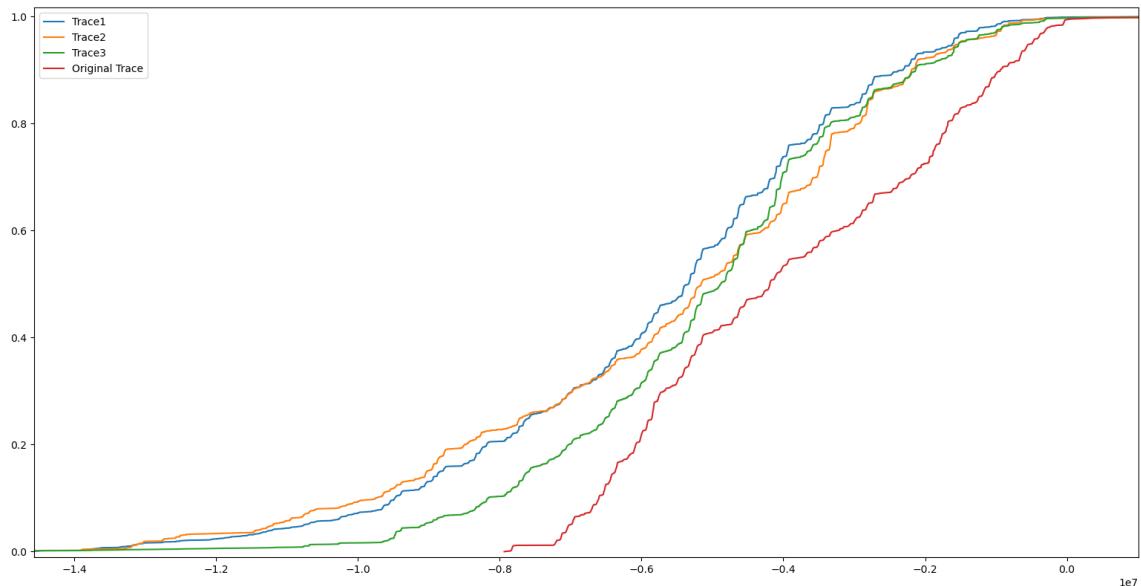


Figure 9-71

Time in seconds / 10^7

PDF Of Think Times with 120% Load

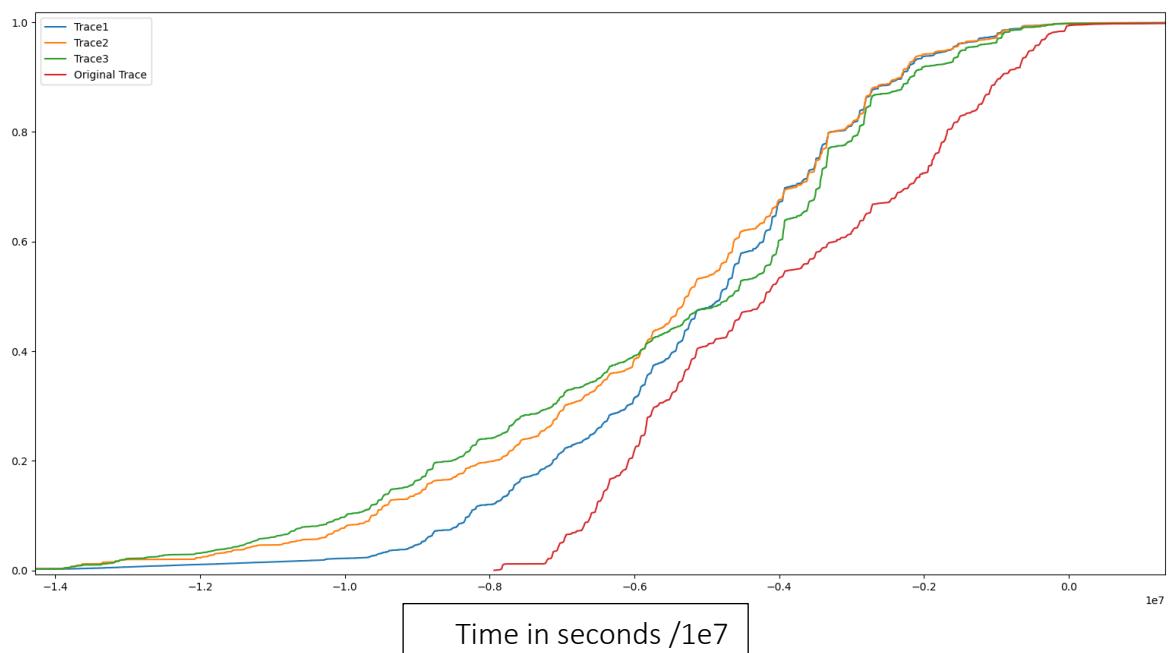


Figure 9-72

- – ביבליוגרפיה ומקורות – The Scientific Python IDE Spyder 4.2.1

MATLAB