Analyzing and Summarizing the RUR Dataset collected from the Titan Supercomputer (2015 - 2019)

Manas P P

Undergraduate at Department of Computer Science and Engineering, Amrita University, Kollam, Kerala, India

1) Problem Statement

In this paper, I work on the first of the four challenges in the Challenge 4 of the SMC Data Challenge 2021, i.e., Analysing Resource Utilization and User behaviour on Titan Supercomputer [1]. We have to perform exploratory data analysis on the dataset. Also a possibility of a relationship between the node count, gpu usage and cpu usage is to be explored.

2) Background and Related work

The RUR (Resource Utilization Report) dataset is collected from the Titan Supercomputer from 2015 – 2019 using the Cray developed resource usage data collection and reporting system. In this paper, I summarize the data characteristics of the RUR dataset, which has usage information of critical resources of Titan and the Science domain in which the job belongs. The work of Wang et al., [2] gives us an overall understanding of the dataset and has analysis complimented by possible causalities for some outcomes.

3) Approach and Uniqueness

In this paper, I try to generate further insights from the dataset. I also work with the science domains and summarize statistics for jobs based on the domain that they belong. I use python and its libraries to work on data analytics as well as data visualization.

Keywords: Titan Supercomputer, Resource Utilization, Data Analysis

4) Findings

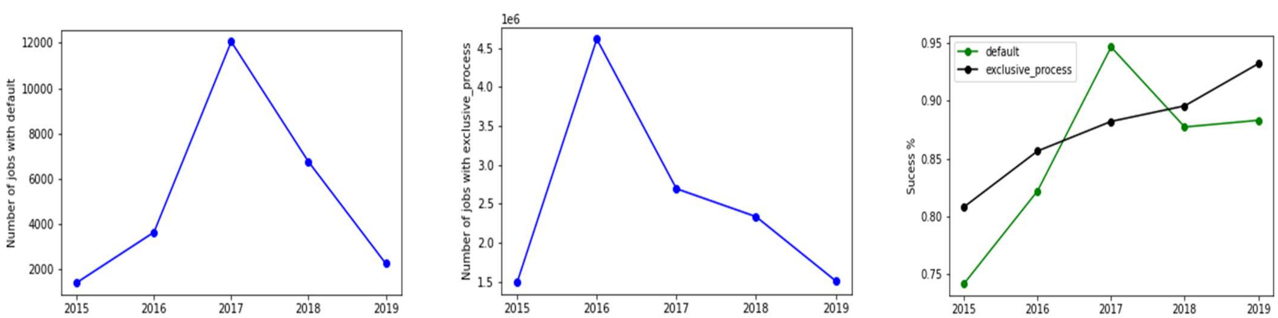


Figure 1 (left) *Figure 1 (centre)* *Figure 1 (right)*

a) The gpu mode “exclusive_process” was clearly favoured over “default” each year. The “default” gpu mode averaged at just 0.00188% of the total jobs over the 5 year period. Hence, overall results like the success percentage over the years was mostly determined by the jobs using the “exclusive_process” mode. In Figure 1(right), the graph shows an increasing success of jobs over the years in the “exclusive_process”. The overall success overlaps (to the naked eye) the line representing the “exclusive_process” and is hence, not shown in the figure. The default mode processes had its best success rate as well as job count (see Figure 1(left) for job count) in 2017 with 94.6% success compared to the 88.2% for “exclusive_process”. The following years of 2018 and 2019 saw a dip in the number of jobs. The peak in 2016 for the total number of jobs was attributed to gpu errors caused due to a manufacturing defect and subsequently led to a smaller but large number of jobs in that year.

b) Over the years, the ratio of tasks with zero gpu usage to the tasks with non-zero gpu usage was always higher than one. The ratio kept increasing each year and in 2019 there was a 12.15 times increase over the previous year 2018 with a value of 90.55 (see Figure 3). The average sum of gpu usage (average of gpu_summem) peaked in 2015 with a value of 54.9 GB per job, the second highest average coming in 2017 with a value of 51.57 GB (see Figure 2). 2016 and 2018 observed much lower average values of 31.9 and 25.25 GB respectively. The year 2019 reported the lowest average gpu usage per job with just 0.181 GB; this can be inferred to the large percentage of jobs done without gpu coupled with the lower number of jobs that year. It has been observed that a job without gpu usage tends to be smaller when compared to jobs with gpu usage. For example, the 75 percentile value of the time span of a job without gpu in 2015 was 40 times less than the corresponding value for jobs with gpu. Table 1 below compares four important characteristics each year.

75 percentile value	Time span(sec)	Max_rss(GB)	Ttime (sec)	Node count
With gpu 2015	5298	1.3	30.9	32
Without gpu 2015	130	0.3	7.2	15
With gpu 2016	3682	2.187	104.24	94
Without gpu 2016	38	0.471	1.428	56
With gpu 2017	3542	0.58	141.30	54
Without gpu 2017	12	0.073	0.062	15
With gpu 2018	3342	0.45	113.05	240
Without gpu 2018	574	0.986	18.12	8
With gpu 2019	5264	5.99	115.33	12
Without gpu 2019	1301	0.192	7.91	2

Table 1

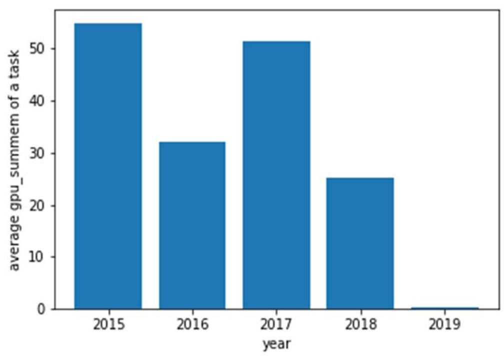


Figure 2

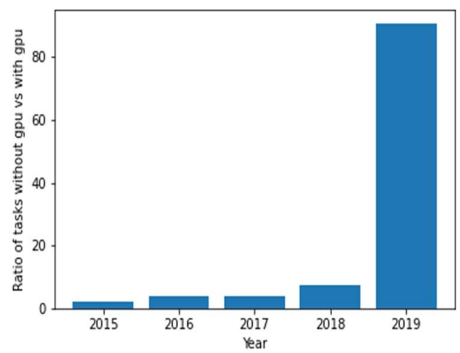


Figure 3

c) We see the average max_rss of the jobs was at the highest point in 2018 with a value of 2.0978 GB, which was at least double of the values in the other years (Figure 5). The average of total time spent by a job on the cpu saw a gradual increase over the years and peaked in 2018 at an average of 667 seconds for a task, which was followed by a decrease to 479 seconds the following year (Figure 4). Although the mean was a few hundred seconds each year, the standard deviations were much higher. For example in 2015, 75 percentile of the jobs had a total cpu time of 10.775 seconds with a standard deviation of 18,429 seconds. The median was 0.07 seconds and the mode 0.000269 seconds, indicating a positively skewed distribution with a median skewness of 0.0757.

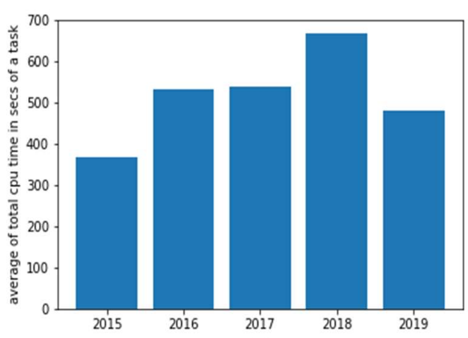


Figure 4

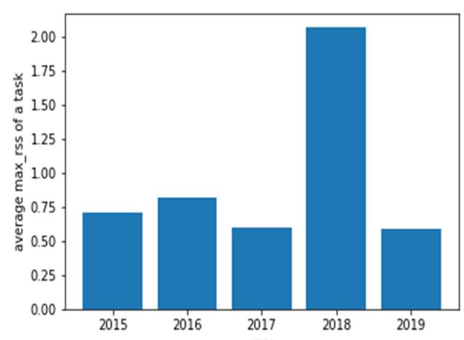
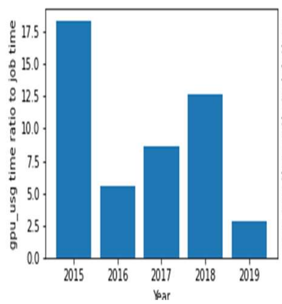
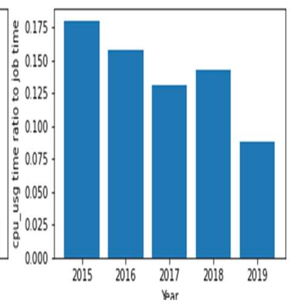


Figure 5

d) Considering the amount of time that was spent on all the gpus by a job (gpu_secs) and taking its ratio with the total job time, we see that 2015 had the highest average ratio of 18.35 and 2018 with 12.6. 2017, 2016 and 2019 had smaller ratios of 8.5, 5.6 and 2.8 respectively (see Fig 6(Left)). Although in Figure 3 we can see that 2018 had a slightly higher ratio of tasks without gpu than with gpu when compared to 2016 and 2017, the jobs using gpu in 2018 were longer with 75 percentile of job time at 732 seconds, while 2016 and 2017 had the same value at 197 and 122 seconds. The same ratio when taken with approximate CPU time (here the base clock of 2.2 GHz was used to calculate the time though a max boost clock of 3.1 Ghz is also possible) shows much lower ratios, with 2015 having the highest ratio of 0.18 while 2019 had just 0.088 (see Fig 6(Right)).





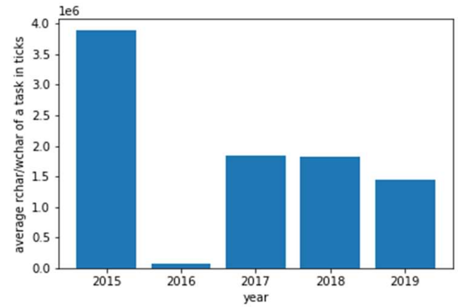


Figure 6(left) *Figure 6(right)* *Figure 7*

e) The ratio of rchar to wchar was a very high value each year, with 2016 having a relatively smaller value when compared to the other years (see Figure 7).In addition, tasks that were successful had a much higher ratio than the tasks that were not successful (see Table 2).

Year	Unsuccessful tasks	Successful tasks	Overall tasks
2015	448,372	4,715,308	3,889,537
2016	71,583	82,783	81,057
2017	84,592	2,074,072	1,835,697
2018	1,13,411	2,012,885	1,813,590
2019	1,80,568	1,538,249	1,445,683

Table 2 Ratio of rchar to wchar

f) The different types of exit conditions (alps_exit, excluding 0) encountered peaked in 2016 with 104 different exit conditions and the following years saw a decline in the type of exit conditions encountered (Figure 8.1). The most frequently occurring exit conditions each year were 0, 137, 1, 127, 143, 139, 2, 130, 255 and 134. The percentage of successful jobs increased over the years; however, 2016 reported the highest number of errors. In Figure 8.3 we see that almost two-thirds of the errors in 2016 were the error codes 127 and 137 with the total errors at 6,62,095.

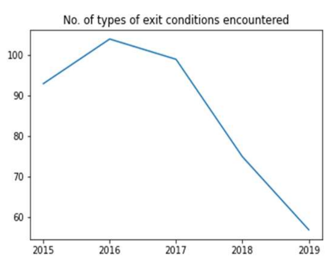


Figure 8.1

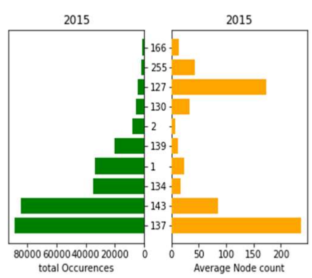


Figure 8.2

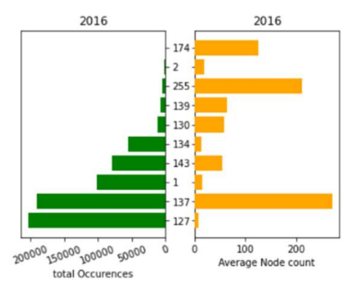


Figure 8.3

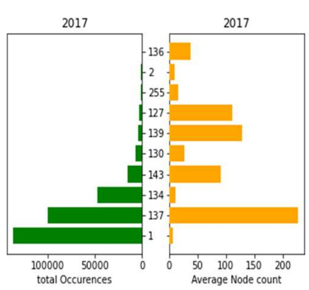


Figure 8.4

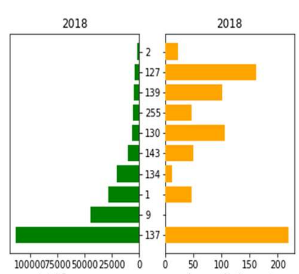


Figure 8.5

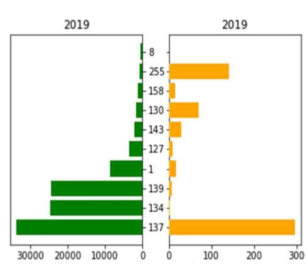


Figure 8.6

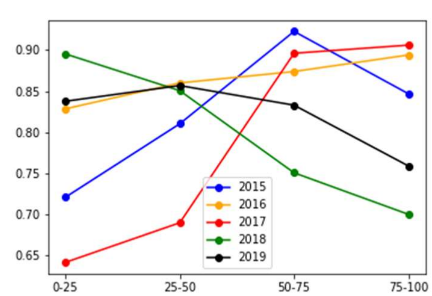


Figure 9, Ratio of users in a percentile to the total number of users that year.

g) The RUR datasets (2015.csv – 2019.csv) were divided into 4 parts in each year according to the job time. The parts had jobs with 0-25, 25-50, 50-75 and 75-100 percentiles of the job time. We observe in Figure 9 that in the year 2015 the ratio of users (unique users) involved in jobs in the 50-75 percentile to the total number of users (unique users) that year was higher than the ratio in the other three divisions. 2016 and 2017 saw the ratio at its highest in the 75-100 percentile range. However, in the years 2018 and 2019 we see that ratio kept decreasing with the higher job time percentiles and 2018 recorded the same ratio at its lowest values for the 50-75 and 75-100 percentiles. The ratio being higher in 75-100 percentile than in the other three divisions meant that there were users that have done larger tasks without a possible trial/testing. 2018 and 2019 saw a higher success percentage of jobs in the 75-100 percentiles than the previous years and we can see the ratio is decreasing with increasing percentile range in 2018 and 2019. Therefore, it is likely that more users have run test models when compared to the previous years of 2016 and 2017. Table 3 below gives the success percentage in each percentile for each year.

Year	0-25 percentile	25-50 percentile	50-75 percentile	75-100 percentile
2015	93.67	75.68	79.77	72.06
2016	72.39	91.31	92.24	88.49
2017	98.32	78.15	83.03	92.06
2018	88.30	81.77	92.63	95.57
2019	89.02	92.94	93.70	97.51

Table 3 (Percentage of successful jobs in each percentile range)

h) The average node count of unsuccessful tasks was higher than the average node count of overall tasks and the average for successful tasks each year.

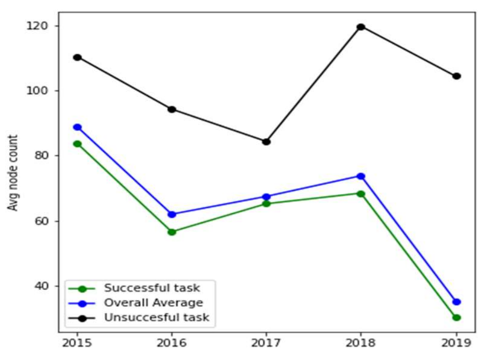


Figure 10, Average node count of unsuccessful, successful and all tasks.

i) Titan saw an increase in the number of users each year, with 2018 having the highest number of users at 850. 2019 however saw a decrease in the number of users, which could be attributed to the launch of Summit in January that year.

Count	2015	2016	2017	2018	2019
Users#	738	793	809	850	419

Table 4

j) The pearson coefficient between log(node count) and log(gpu_summem) for a task with non-zero gpu_usage was 0.8889 in 2017 (see Table 5.1). A linear model fit with x = log(node_count) and the target y as log(gpu_summem) gave a best fit line with a slope of 1.143 (Figure 11.1). This model has an adjusted-R-square value of 0.7433, 0.7474 and 0.7648 for the score of prediction for the years 2015, 2016 and 2018. However a poor score of -0.449 was encountered in 2019, and the subsequent correlation of the two variables in the year 2019 turned out to be a just 0.4675. 2019 had a large number of jobs with zero gpu and the graph for jobs with gpu usage was highly dispersed.

Year	Correlation	adjusted-R squares
2015	0.86771	0.7433
2016	0.87246	0.7474
2017	0.88946	-
2018	0.88958	0.7648
2019	0.46758	-0.4768

Table 5.1 Correlation (between log (node_count) and log (gpu_summem)) and adjusted-R fit (linear model fit on 2015 data)

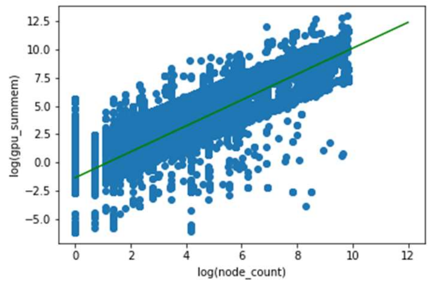
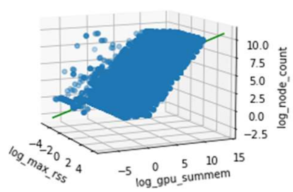


Figure 11.1 log (gpu_summem) vs log (node_count)

k) Further two way analysis of characteristics based on node count, CPU usage and GPU usage did not yield satisfactory results. However, a three way analysis of log(max_rss), log(gpu_summem) and log(node_count) of jobs with non-zero gpu usage yielded a linear model with log(max_rss) and log(gpu_summem) as the independent variables and the target variable as log(node_count) that had adjusted-R square values as shown in Table 5.2. Data from 2015 was used to fit the line, and the coefficients of fit were [-0.4846 , 0.8596] (Figure 11.2). Just like in point j) 2019 had a poor fit.



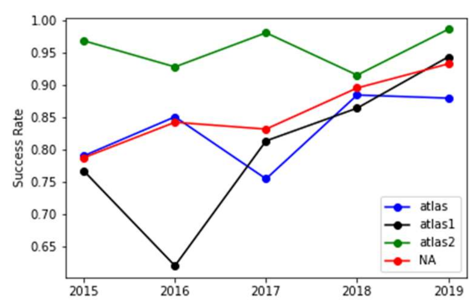
Year	adjusted-R squares
2016	0.831
2017	0.844
2018	0.855
2019	-2.447

On the left Figure 11.2

On the right Table 5.2

l) Among the jobs which had a science department assigned to it, the file_system had three categories atlas, atlas1 and atlas2. In 2017 there was another category encountered, the atlas1_thin (Table 6). Atlas2 has the highest success percentage over the years with at least 91% each year and an impressive 98.67 % in 2019. Atlas1_thin had only 254 jobs in 2017 and recorded a success percentage of 85.433% (Figure 12).

Figure 12 ratio of successful to total jobs



In 1000s	2015	2016	2017	2018	2019
NA	895.9	3359.8	1411.6	1737.7	1229.1
atlas	404.9	430.7	236.7	317.6	149.8
atlas1	25.9	30.6	14.8	42.2	9.8
atlas2	165.2	795.7	1040.9	241.7	121.5
atlas1_thin	0	0	0.254	0	0

Table 6 Count of jobs in each file system in 1000s

m) Considering the ratio of time spent in user mode to time spent in kernel mode of the jobs, we see that after 2016 the average of this ratio was higher for successful jobs when compared to unsuccessful jobs.

Year	Avg ratio of unsuccessful task	Avg ratio of successful task
2015	58.289	37.242
2016	66.395	44.730
2017	36.761	61.277
2018	43.138	124.681
2019	36.509	394.969

Table 7 Ratio of time spent in user mode to time spent in kernel mode.

Year	Utime/stime < 1	Utime/stime > 1
2015	1838.657	4276.664
2016	76.380	4148.029
2017	211.662	2674.864
2018	90.751	3024.673
2019	76.557	4223.082

Table 8 Average job time (only jobs with science domain) in seconds each year for the two categories.

n) The jobs in all the sciences over the 5 year period saw jobs with a higher utime than stime having a higher average job time of 3669 seconds when compared to 458.8 seconds when the ratio was less than 1, showing a greater reliance on user mode for larger jobs (Table 8).

o) Taking individual users and their success in that year, we try to fit a curve that best describes the estimated number of users that has a particular success percent in that year. We use the histogram data, i.e., the bin locations (ratio of success) and data entries (number of users) and fit a curve using least squares fit. The function used for the curve was

$$F(x) = (a * e^{(b*x)}) + c$$

We see from the curve fits that 2016 and 2017 had similar curves that gradually increased while 2018 and 2019 had curves that were similar to each other with a steeper rise from 0.8 onwards.

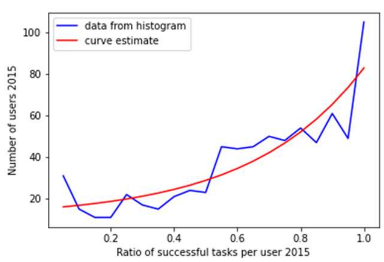


Figure 13.1

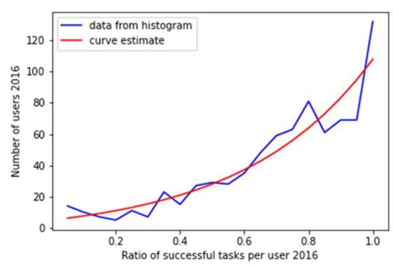


Figure 13.2

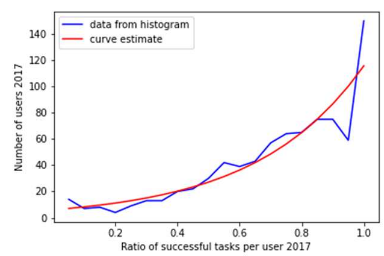


Figure 13.3

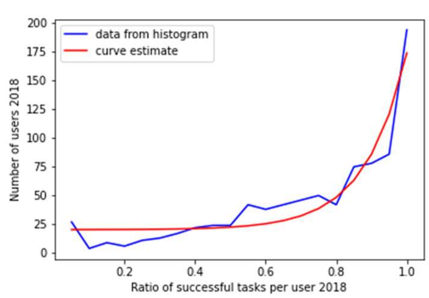


Figure 13.4

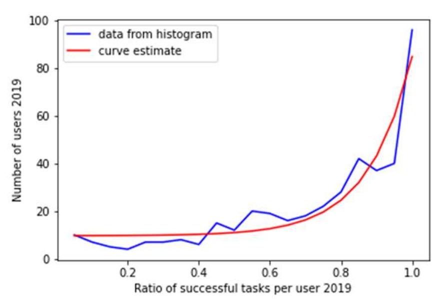


Figure 13.5

The values of a, b and c over the years are shown in Table 9.

Year	a	b	c
2015	4.40	2.79	11.02
2016	9.33	2.49	-4.33
2017	6.54	2.87	-0.46
2018	0.03	8.50	20.32
2019	0.02	8.09	9.66

Table 9, the parameter values for each year's curve

Science	2015	2016	2017	2018	2019
Biology	62.37	50.97	50.85	82.16	69.62
Chemistry	58	-	42.74	51.31	89.36
CS(computer science)	91.74	93.93	97.73	89.67	97.39
Earth Science	90	93.38	87.12	95.32	98.24
Engineering	62.94	68.39	79.74	88.85	95.51
Fusion	92.53	86.60	80.24	95.97	96.15
Materials	59.99	84.34	95.05	92.69	88.96
Nuclear	59.15	79.56	73.88	68.93	100
Physics	93.46	70.66	35.35	60	67.88

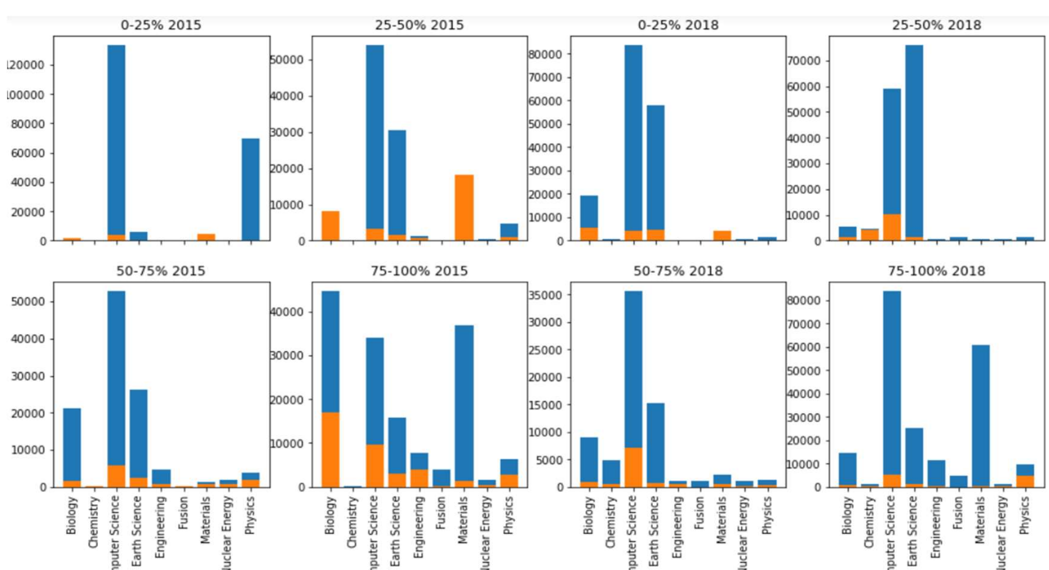
Table 10 the percentage of successful jobs in each year for a particular science domain.

p) There were nine fields of Science in the RUR dataset. Each field further has areas in them, which made up 29 in total. These fields enjoyed variable levels of success, which is shown in the Table 10. No job in chemistry was encountered in the year 2016. The command column also had an ID for each job under the science. Table 11 below shows the least successful IDs and the area of science associated with them.

2015			2016			2017		
ID	**Area**	**Success %**	**ID**	**Area**	**Success%**	**ID**	**Area**	**Success%**
Area50	Systems Biology	0	**Area31**	Nano Electronics	22.5	**Area34**	Nuclear Physics	30.1
Area38	Nano- Science	34.7	**Area38**	Nano Science	25	**Area60**	Vendor	39.6
Area55	Turbulence	35.7	**Area33**	Nuclear Fuel Cycle	35.5	**Area8**	Biophysics	41.2
Area30	Atomic Physics	38.6	**Area8**	Biophysics	40.4	**Area10**	Chemistry	42.7
Area13	Combustion	52.4	**Area30**	Atomic Physics	42.2	**Area29**	Medical Science	45.3

2018			2019		
ID	**Area**	**Success%**	**ID**	**Area**	**Success%**
Area55	Turbulence	9.4	**Area22**	General (CS)	8.5
Area60	Vendor	21.6	**Area29**	Medical Science	50.7
Area11	Condensed Matter Physics	32.8	**Area24**	High Energy Physics	51.6
Area34	Nuclear Physics	33.5	**Area8**	Biophysics	55.5
Area20	Enivronmental Science	46.6	**Area2**	Aerodynamics	58.1

Table 11



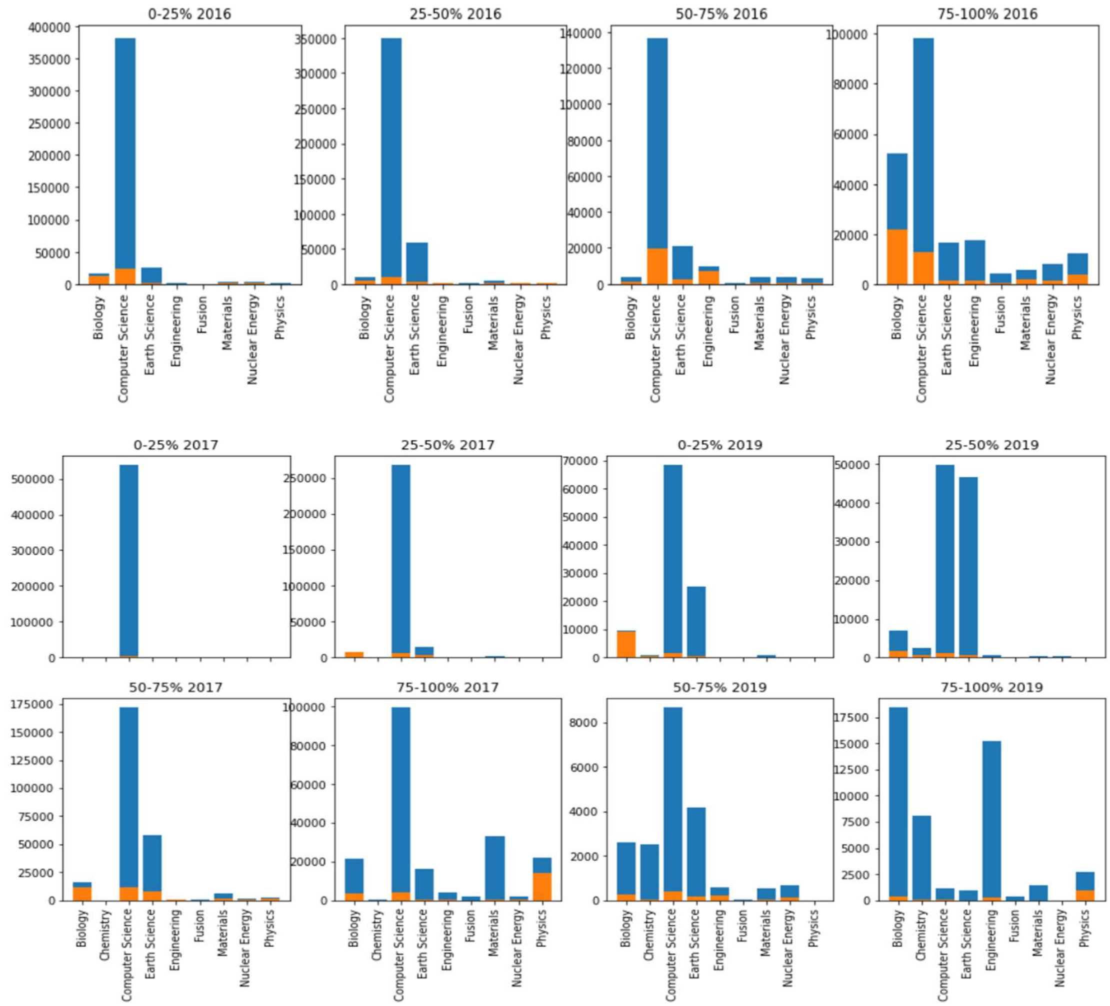


Figure 14 Bar graph of number of jobs in each percentile (same divisions as in point g) in a year based on science domain (blue) and the corresponding number of failed jobs (orange).

5) Refrences

Datasets : https://doi.ccs.ornl.gov/ui/doi/334

[1] Analyzing Resource Utilization and User Behaviour on Titan Supercomputer, Sajal et al,. DOI : 10.13139/OLCF/1772811

Support for 10.13139/OLCF/1772811 dataset is provided by the U.S. Department of Energy, project GEN150 under Contract DE-AC05-00OR22725. Project GEN150 used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

[2] Learning from Five-year Resource-Utilization Data of Titan System, Wang et al..

Appendix

Due to a lengthy code, I went with a github link for the code.

Link to the codes: https://github.com/Manas641/RUR-Dataset-Analysis