REDUCTION OF DAILY COMPUTE USAGE

Kyle Wojtaszek – MBC 638

DMAIC Storyboard

Define
4/15-4/28

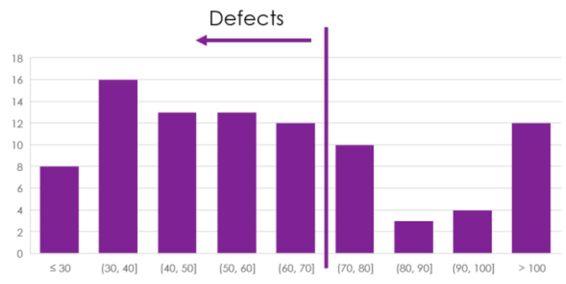
- Problem
  - Daily script workload uses a lot of compute power to process our incoming data. Lowering this amount directly saves the company money.
  - 'Defect' defined as a job with a resource usage efficiency **less than 70%**
- Goal
  - Increase SQL to 1.50

Baseline Efficiency Descriptive Statistics	
Median	55%
Average	65%
Minimum	22%
Maximum	199%
Standard Deviation	35.92%
Margin of Error	7.38%

Measure
5/1-5/14

- Data Collected
- Baseline established
- Initial SQL calculated

Current SQL
1.03



Analyze
5/10-5/21

- Regression

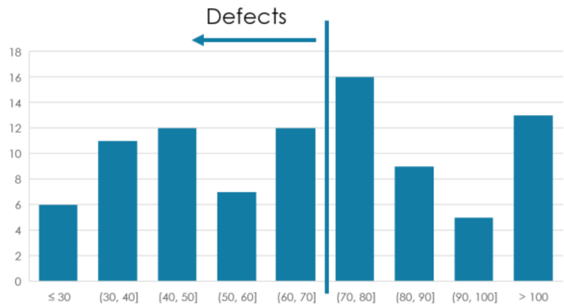
Regression Statistics	
Multiple R	0.90847487
R Square	0.82532659
Adjusted R Square	0.817202245
Significance F	9.50418363E-32
Observations	91

- **Vertex initialization costs** and **low concurrency problems** are hurting efficiency

Improve
5/28-6/1

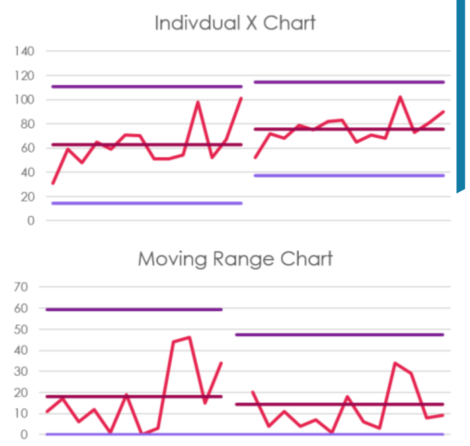
- Action Plan
  - **Decrease vertex initialization costs** and **increase concurrency**

New SQL
1.43



Control
6/1-Current

- Continue to monitor SQL
- Control charts to protect against regressions



PROBLEM STATEMENT AND OBJECTIVE

Define

- Every day our team runs automated scripts to process incoming telemetry data
  - These scripts are ran on a central computing cluster and each unit of computation costs the company money
- We want to reduce the amount of computational power used, while still maintaining the business value of our automated scripts

- Key Metric: Job Efficiency
  - A metric provided by the cluster computing system which tells us how efficiently we are using the computing resources assigned to us.
  - Measure in percentage, ranges between 0-200%
  - Will be used to calculate SQL of our automated scripts with a “defect” being defined as any script below 70% efficiency

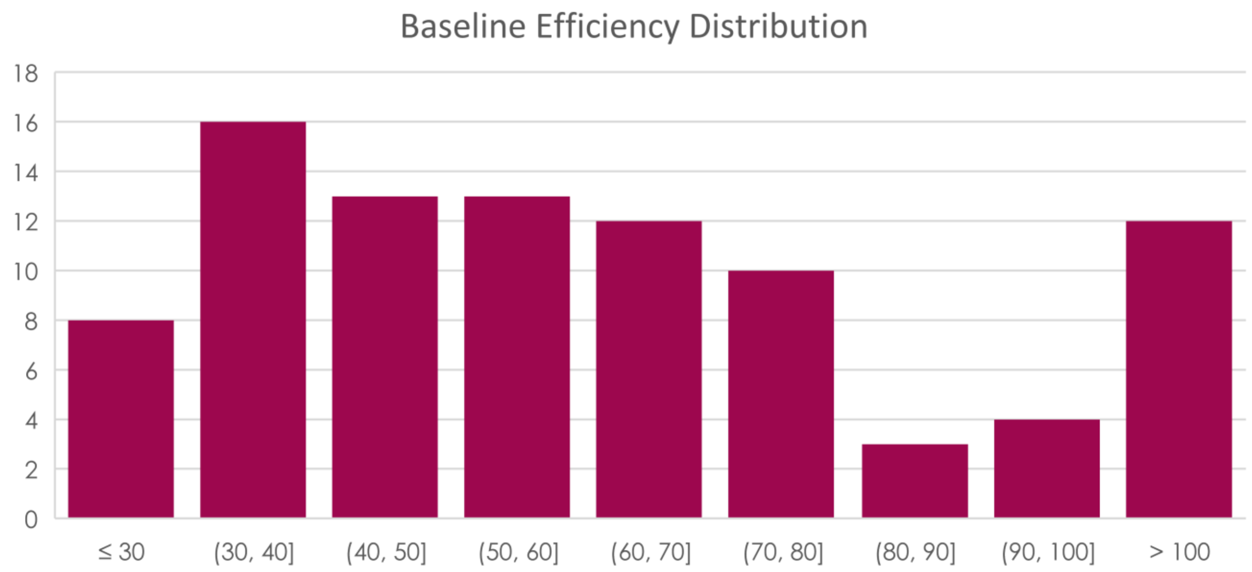
- Objective: Increase SQL of the process from initial value of 1.03 to at least 1.50

Define

SCOPE AND BASELINE

- Data will be collected for a set of 13 scripts which are run daily and contain the majority of our important business logic
    - One-off experimental scripts and any new scripts onboarded during the collection period will not be measured
    - Execution from days where there are system-wide technical issues will also be ignored as these issues are not in our control

Baseline Descriptive Statistics	
Sample Size	91
Median	55%
Average	65%
Minimum	22%
Maximum	199%
Standard Deviation	35.92%
Margin of Error	7.38%



DATA MEASUREMENT PLAN

Measure

Performance Measure	Data Source and Location	How Will Data Be Collected	Who Will Collect?	When?	Target Sample Size
Job tokens used	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Job Execution Time	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Job Compute Hours	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Job Data Size	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Execution Density	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Job Efficiency	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Low Concurrency Token Percent	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Job Initialization Cost	Job results	Pull from report	Kyle	5/1-5/14	7-14 days
Vertex Initialization Cost	Job results	Pull from report	Kyle	5/1-5/14	7-14 days

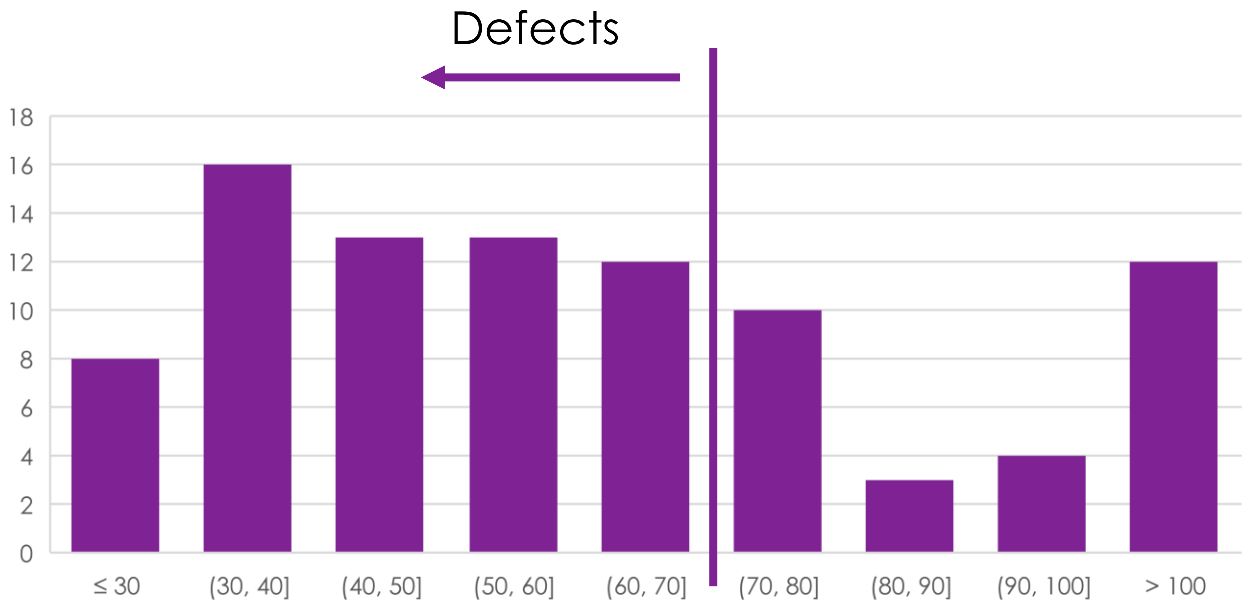
Data measurement to take place over 14 days. Some days had system failures beyond our control, this resulted in 7 days of good data collected across 13 different scripts.

Measure

SIGMA QUALITY LEVEL

- Defect defined as any script which has an efficiency of less than 70%
- Below is the histogram from our Define phase which includes all of our measurements with an arrow added to denote where the defects lie
- 91 total scripts run with a total of 62 defects
    - DPMO: ~681,000

SQL: 1.03



IS JOB EFFICIENCY A GOOD MEASURE?

Measure

- Given that the scripts are not changing day to day, we should expect job efficiency measurements to be **reproducible** on a given script.
  - Why not **repeatable**? Due to the nature of cloud computing, the scripts may be run on different hardware day-to-day. Since the hardware is the 'operator' or our measurements and may have varying specs and bandwidths, we consider this to be a different operator each day.
- Efficiency metric is continuous, however we need discrete data to measure Kappa
- To achieve discrete data, we can categorize each continuous result as a defect or not, based on our SQL measurement's definition
- We will measure this across our 13 scripts between two different days

	Good	Bad
Day 1	4/13	9/13
Day 2	3/13	10/13

P(Observed)	12/13
P(Chance)	0.60355
Kappa	0.80597

Kappa Level of **0.8** indicates that our measurement system is **pretty good**

Repeating the Kappa calculation against other pairs of days gave similar results.

MULTIPLE REGRESSION: DEFINITIONS

Analyze

- In addition to efficiency, I've collected additional data about each job execution. These are properties of the job which may or may not contribute to the efficiency
- We hope to find potential relationships via multiple regression
- These additional data points are:
  - Job tokens used – The amount of computing resources allocated to the job
  - Execution time – Real-world time for job completion
  - Compute time – Amount of total compute time used
    - Cluster computing is highly parallel, so for instance if you are doing 10 things at once for 1 real-world minute, that is 10 minutes of compute time)
  - Job Initialization – Percent of resources spent initializing the job
  - Vertex Initialization – Percent of resources spent starting up compute vertices
  - Low concurrency density – Percent of resources wasted by idling because job is not parallel enough
  - Execution density – Percent of resources actually used for computation.
  - Input Data Size – Amount of data read in by the script

MULTIPLE REGRESSION: RESULTS

Analyze

- After running a few different regression passes, it was determined that the following fields we're **not** relevant to the regression, given their high p-values:
  - Job Tokens Used, Execution Time, Job Initialization, and Input Data Size

- Multiple Regression Equation:

  Efficiency = 22.25 + 17.04 * (Compute Hours) + 2.57 * (Execution Density) – 0.42 * (Vertex Initialization) – 0.49 * (Low Concurrency Density)

- F is low, meaning at least one x is significant
- R of 0.9 indicates a high correlation
- Adjusted R^2 indicates we are explaining 81.7% of efficiency in this model

Regression Statistics	
Multiple R	0.90847487
R Square	0.82532659
Adjusted R Square	0.817202245
Significance F	9.50418363E-32
Observations	91

MULTIPLE REGRESSION: ANALYSIS

Improve

- Model indicates that Vertex Initialization and Low Concurrency Density are two of the factors which bring down job efficiency.
- Action Plan:
  - Identify scripts with highest Vertex Initialization and Low Concurrency Density
  - Work with team to improve these metrics
  - Re-measure to ensure improvements in overall job efficiency

ACTION PLAN RESULTS

Improve

- Two underperforming scripts were identified using the metrics from our Action Plan
- We were able to refactor the scripts to increase concurrency and decrease vertex initialization costs.
- New data was acquired for these two scripts during the week of May 28th

- Hypothesis: Increasing the concurrency and decreasing the vertex initialization costs will cause a statistically significant increase in job efficiency.

Improve

HYPOTHESIS TESTING

- Our data is continuous and we have two samples
- Each sample has 7 days of data for two scripts, totaling 28 samples. Since this is less than 30 samples, we must used the t-test
- We will be calculating a lower/left tail p-value with the following hypotheses:

$$H_0 : \text{Efficiency}_1 >= \text{Efficiency}_2$$

$$H_a : \text{Efficiency}_1 < \text{Efficiency}_2$$

	Before	After
Mean	62.64286	75.78571
Standard Deviation	18.75185	11.9819
Count	14	14

Results	
t-Score	-2.20986
df	25
p-value	0.0175
alpha	0.05

Based on these results:

Our p-value is lower than our alpha, which means we can safely reject the null hypothesis (H_0) and say that we have **successfully increased the efficiency of our scripts**.

Control

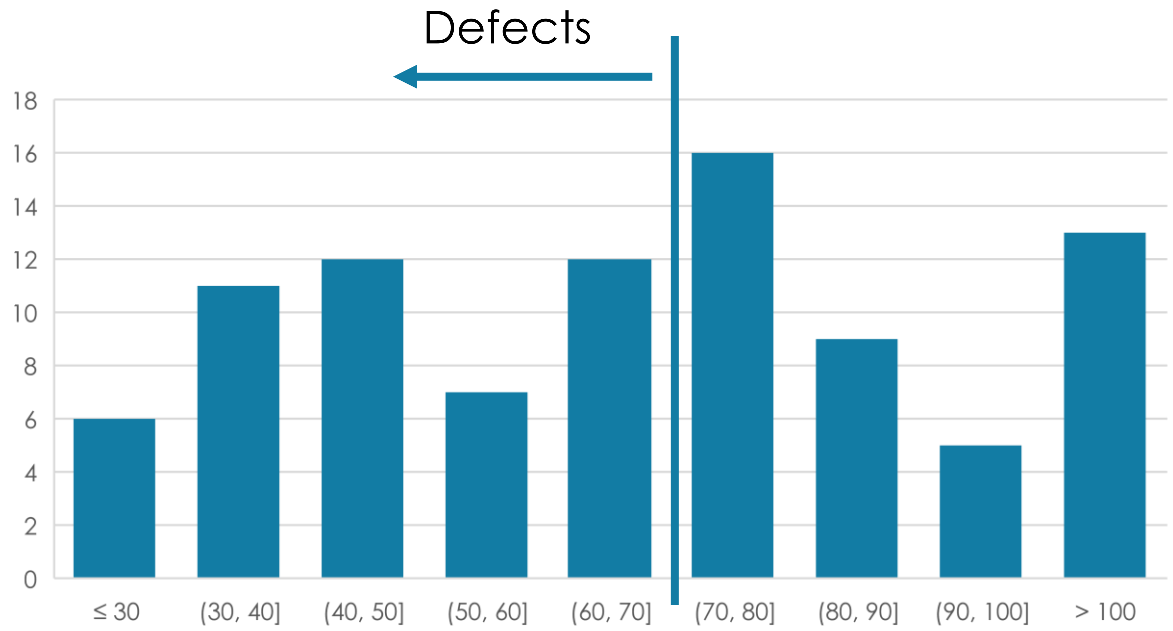
NEW SIGMA QUALITY LEVEL

- Using our analysis, we applied our newfound knowledge across other under-performing scripts and recollected data from June 6^{th} to June 13^{th} in order to determine our new SQL.
- 91 total scripts run with a total of only 48 defects
  - DPMO: ~527,000

SQL: 1.43

We didn't quite meet our original goal of 1.50.

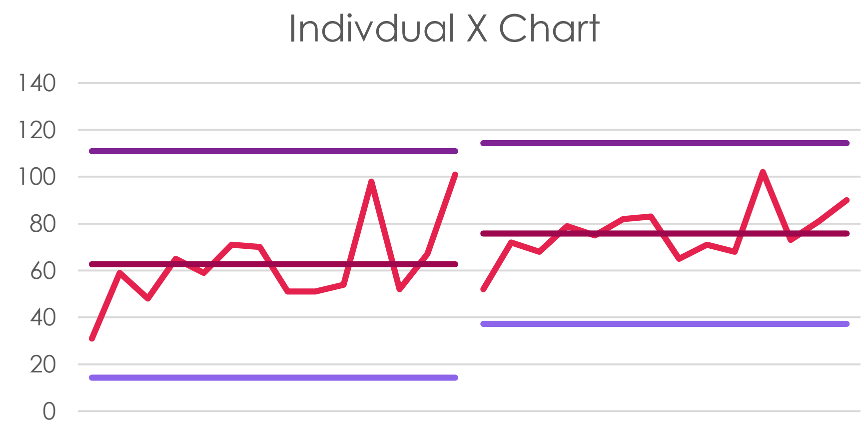
However, we did see a significant increase and now have a good basis for future improvements.

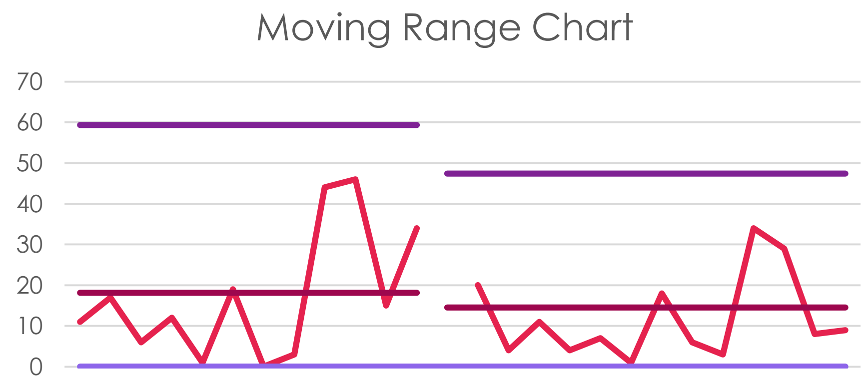


Control

CONTROL CHARTS

- In order to monitor day-to-day changes and ensure that our efficiency stays within control, a control chart will be created for each script
  - Each script has unique characteristics, thus requiring each script to have its own chart, as opposed to one chart for all of them combined
- To the right is the chart for one of the scripts which we improved based on our regression. You can see the clear break where efficiency went up in the X chart and range went down in the Moving Range Chart
  - This indicates that our changes had a positive effect on our process.





FUTURE CONTROL

Control

- We will continue to monitor SQL and expand this measurement to all of our scripts
- We will monitor the control charts to ensure there are no regressions in efficiency
- There are still some scripts that can be optimized, each optimization is time consuming and we were not able to complete them all before the project due date.
- Continued optimizations should push us to our goal of an SQL of 1.50.