NetApp Autosupport Analysis

Junwei Da

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-158
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-158.html

June 2, 2012

Acknowledgement

I'd like to thank my teammates Shubh Manjayya and Muni Xu for there contribution in the project. I'd also like to thank my faculty advisor Professor Ikhlaq Sidhu, who has provided guidance and project management. I'd like to thank Yanpei Chen, our Ph.D advisor, whose technical insights and experience always breathe new ideas into our project. Last but not least, I'd like to thank Satish Penmetsa, who has got me interested in the project from the start, as well as being the high level thinker of the project; his encouragement from the industry side has given us much confidence in the value of the project.

NetApp Autosupport Analysis

Master of Engineering

Capstone Project Report

Junwei Da

UC Berkeley

Abstract:

Big data has taken the tech industry by storm as storage costs go down and analytics tools improve to enable businesses to make better decisions faster. NetApp is one such company that collects customer machine configurations through NetApp Autosupport to help customers troubleshoot errors. This project leverages the Autosupport data to gain insights into the production environment as well as the QA environment in terms of their relationships to each other. Using the K-Means algorithm and direct matching method, we have identified eight common customer configuration groups, top customer configurations not tested by any QA machines, and top QA machines not testing any customer configurations. The methodology is still maturing, and requires input from both developers and subject area experts. The results we found can be used to enhance the test environment for QA, target development of features for developers, and increase confidence in product and services for customers.

I. Introduction:

The success of web 2.0 era is partially attributed to the appropriate use of data driven applications. User data is collected in structured format from the Internet before being analyzed and organized to enable a number of other applications. Data is used not only for transactional purposes, but also for tracking user behavior and making predictions before underlying trends become noticeable to the community at large. With the emerging era of ubiquitous computing, enabled by mobile smart phones in everyone's pocket and cloud computing in big data centers, more data has been collected in unimaginable ways. However, as we collect more data, the need for structured data has become restrictive rather than empowering because data is evolving at a rate that structured database can no longer keep up. As a result, much of the unstructured data were stored in flat file format, waiting to be analyzed and make sense of.

Within the myriad of data collected embeds useful information that reflects behaviors of its source and the world at large. Data mining is “the science of extracting useful knowledge from [unprecedented quantities of digital data] repositories”, and it is quickly gaining popularity as many believe that it “can bring real value”. [1] Businesses can take advantage of the information extracted from data to gain insights into their business operations and take necessary actions to meet new demands.

NetApp is a storage vendor that sells configurable storage filers to technology companies, financial institutions, government agencies, etc. NetApp Autosupport is “an integrated and efficient monitoring and reporting feature that constantly checks the health of [the filer] system”. For each of the customer machines, Autosupport constantly sends the user configuration information back to NetApp, where the information is analyzed and stored to keep a history of the systems health record. Within the company, NetApp has many similar machines set up internally for the purpose of quality assurance (QA), both to test new features and to replicate reported customer problems. In this paper, we use the customer data collected by Autosupport and the data about QA machines to answer the question of whether the current NetApp internal testing machines represent a good coverage of the customer machines in production. Our methodology has taken two data mining approaches, one through a simple direct matching of data attributes, and one through the statistical K-Means algorithm. We have identified:

- Eight common customer configuration groups
- Top customer configurations not tested by any QA machines.
- Top QA machines not testing any customer configurations.

Given the above knowledge, NetApp can develop features targeted at each customer group, and enhance their QA testing environment by adding QA machines to cover untested customer configurations and retiring unnecessary QA machines to optimize resource efficiency. In the rest of this paper, we present our analysis in detail by researching existing approaches to similar problems (section II) before defining our problem set up (section III). For our methodology, we first propose a simple yet intuitive approach using direct matching analysis and present its results (section IV). After we identify some shortcomings of the first approach, we investigate further by applying the K-Means algorithm to analyze the dataset further (section V). We then validate the second approach by presenting our analysis results and discuss their implications (section VI), demonstrating the insights they reveal about the QA environment in relation to the systems in production. Lastly, we discuss opportunities and challenges for future work (section VII).

II. Literature Review/Related Work

Similar researches have been done in the past where large data set has been analyzed for clustering pattern. Finding out the clustering pattern helps to gain a summarized overview of the entire data set, as well as how different types of data distribute within the entire set.

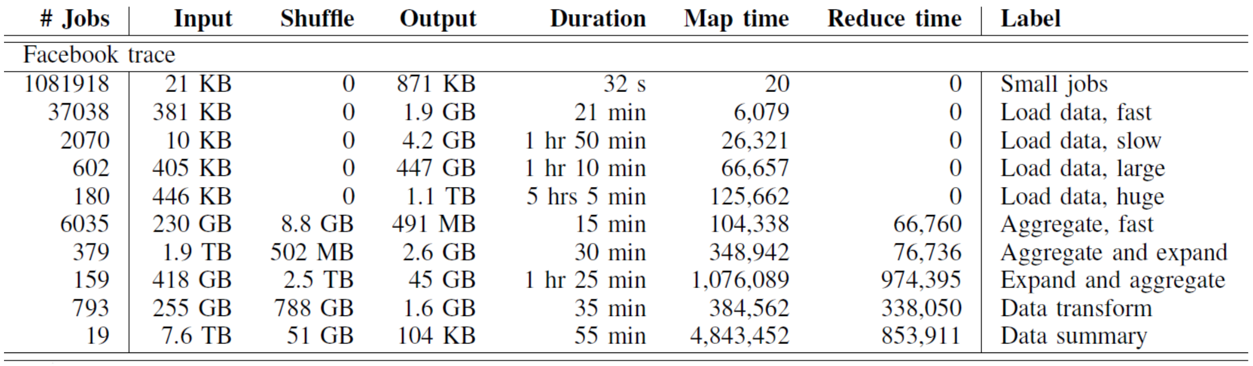
Classification of MapReduce Workloads

Hadoop is a popular MapReduce framework used by many large internet companies. The software framework is used to batch process large amounts of unstructured data such as server logs and sensor measurements in a distributed manner that also has fault-tolerance built-in. The typical set up is many compute nodes interconnected together to become a distributed file system managed by Hadoop, and the data nodes and master node together forms a Hadoop cluster. A MapReduce job is a program that is applied to files on different compute nodes and can range from small to large depending on the type of operation and the input size of the file. [2]

However, despite its widespread adoption, there are still many aspects of Hadoop that are not optimized for the production work it is give. For example, if there are a series of small jobs to be run after a big job, the big job would often become the bottleneck of the entire Hadoop cluster. Studies have been done on the type of jobs that are given to the Hadoop clusters as different scheduling schemes would accommodate different Hadoop job types.

The K-means clustering analysis was used to find out the classifications of the different Hadoop job types, as part of a research conducted by Chen from UC Berkeley. [3] The study analyzed a trace, a history containing meta-data about the individual Hadoop jobs, with duration of one year from Facebook. Seven different attributes, indicative of the input job type, were used to characterize Hadoop jobs in the Facebook trace, which are input size, shuffle size, output size, total execution time, map time, and reduce time. The parameters were linearly normalized such that all data dimensions have a range between 0 and 1 to account for the different units. The number of clusters, k, was incremented to show the quality of cluster assignment of points. The cluster centers are shown in table 1 in terms of the seven dimensions they have.

Table 1 Cluster Centers for the Facebook Trace [3]



# Jobs	Input	Shuffle	Output	Duration	Map time	Reduce time	Label
Facebook trace							
1081918	21 KB	0	871 KB	32 s	20	0	Small jobs
37038	381 KB	0	1.9 GB	21 min	6,079	0	Load data, fast
2070	10 KB	0	4.2 GB	1 hr 50 min	26,321	0	Load data, slow
602	405 KB	0	447 GB	1 hr 10 min	66,657	0	Load data, large
180	446 KB	0	1.1 TB	5 hrs 5 min	125,662	0	Load data, huge
6035	230 GB	8.8 GB	491 MB	15 min	104,338	66,760	Aggregate, fast
379	1.9 TB	502 MB	2.6 GB	30 min	348,942	76,736	Aggregate and expand
159	418 GB	2.5 TB	45 GB	1 hr 25 min	1,076,089	974,395	Expand and aggregate
793	255 GB	788 GB	1.6 GB	35 min	384,562	338,050	Data transform
19	7.6 TB	51 GB	104 KB	55 min	4,843,452	853,911	Data summary

Further analysis of the cluster centers by subject area experts revealed the 10 common classifications of the MapReduce jobs, and meaningful labels were attached to each class as shown in the last column. Now that one have identified the different MapReduce job types, optimization could be done that would differentiate the scheduling and resource allocations for the jobs, instead of treating all the jobs in the same way. The K-means algorithm has revealed important characteristics of the MapReduce workload for Facebook that will improve Hadoop operations in the future. [3]

Summary of How They Informed Our Project

The case study of the MapReduce job classification has given us confirmation in the K-means clustering analysis method. It has a similar aspect to our project in that we need to transform our dataset into meaningful dimensions that would reflect the most information about the clusters. In addition, we have discovered a parameter for measuring the quality of a clustering assignment, percent variance explained, which would help us decide what would be the natural grouping of the clustering points after running the algorithm with different given k.

III. Background – Autosupport vs. QA Data

Product testing at NetApp has many phases depending on the release life cycle of existing and new product. The phases include [4]:

- Planning for the new product.
- Development and unit testing before check-in.
- Integration Acceptance Test: test to ensure parts of the product work with each other
- Feature Function Test: feature testing, regression testing against current and previous issues, and performance testing to ensure product operates at defined state.
- Regression and Reliability Test: full product tests after issue/fix phase to bring the release to release candidate state.

The major testing phase is the feature function test, where the quality assurance (QA) team thoroughly tests the product according to the functional specifications and looks for any potential security vulnerabilities. [5] Because NetApp sells hardware as well as the software that runs on top of the hardware, the testing environment must include the two levels as well.

For the particular set of features to be released into production, QA will configure specific parameters on physical hardware systems in addition to tuning the software parameters before feature testing could take place. The number of permutations of all the parameters is simply too large for QA to test exhaustively. [6] As a result, QA can only test a subset of all possible set of configuration parameters before the product is released into production. Since QA machines are set up based on the new products that are released and when particular customer problems are reported, NetApp has no knowledge whether their test environment is representative of the customer machine out in the real world. It is almost guaranteed that a customer of the product will be using the released features under a set of configurations not tested by QA.

Because a storage filer has thousands of configuration parameters, it is infeasible and unnecessary to consider all of them. With help from industry experts, we have identified 21 parameters that would provide the most insight to identify a system. Of these we picked eight parameters to analyze for similarities between the QA machines and customer machines. They are System Model, System Version, Total Volume Count, Total Aggregate Size (GB), Disk Total Size (GB), CIFS Licensed, NFS Licensed, and FCP Licensed. For our project, we received two data sets, one for customer machines and one for QA test machines, which contain configuration parameters for each machine. The customer configuration data comes from Autosupport messages sent by customer machines, and the test machine data comes from the QA team. The customer data contains information for 133,069 customer machines, and the QA data contains information for 1865 test machines. The project is to find out how well the test machines represent the customer machines, and the customer segments using NetApp products.

IV. Methodology: Approach 1 – Direct Matching

One simple approach to find out how well the QA data represents the customer data is through direct matching of the configuration parameters. Given the eight configuration parameters, we find out unique configuration types from the customer data in terms of the exact value of a parameter, and determine which of the QA machine configurations matches the unique configuration types exactly.

Since direct matching may not be possible on continuous values such as storage size, binning is used where ranges of values have been used for matching.

Using this method we have identified over 1400 unique customer configuration types and the QA machines with similar configurations. Figure 1 shows the cumulative percentage of all the configuration types. From the graph, we can see up to 75% of the customer configurations have been covered, represented by blue circles, and the red circles represent the untested customer configurations.

The direct matching method, given its simple and quick solution, is not scalable for a large number of parameters. This is because as one increase the dimensions, the number of permutations will also increase significantly, making exact matches nearly impossible. This weakness could be addressed in the K-Means analysis, where all dimensions are taken into account for determining closeness between two machine configurations.

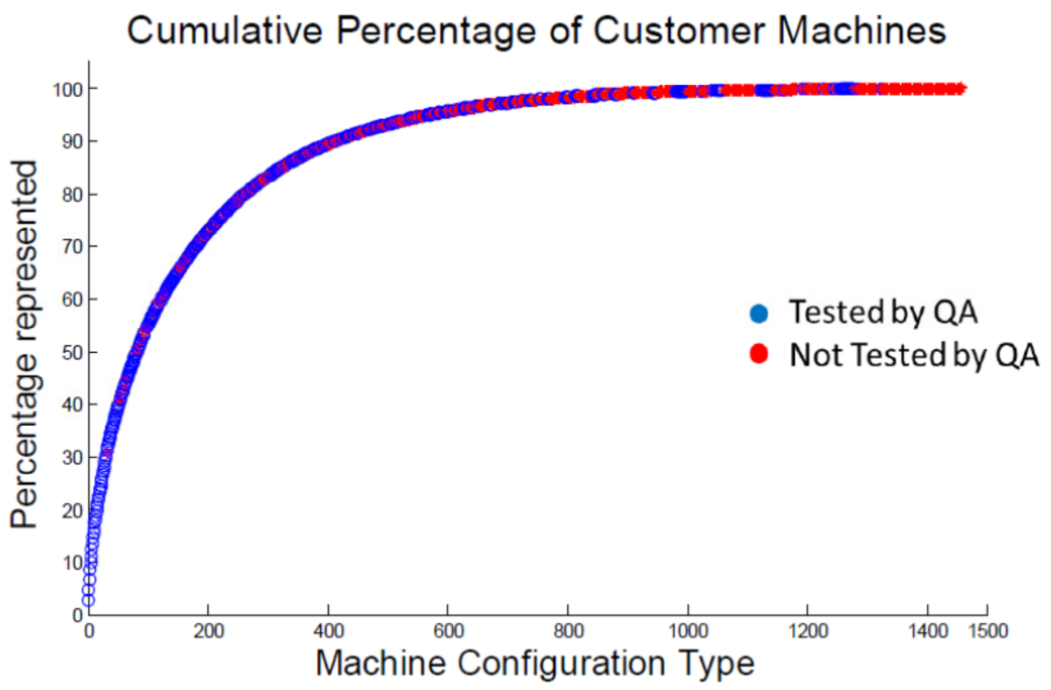


Figure 1 Cumulative Percentage of QA Coverage

V. Methodology: Approach 2 – K-Means Algorithm

The K-Means algorithm, as discussed earlier in literature review, is a statistical analysis method that finds the natural clustering of a data set. What the algorithm does is that it considers each data point to be a N-dimensional tuple, and determines the closeness between two points by their Euclidean distance in N-dimensional space. [7] This property suits well with our problem as the description of a machine configuration is a multi-dimension array where each dimension corresponds to a configuration parameter.

The K-Means algorithm is demonstrated in figure 2. It is a stochastic and recursive algorithm where *n* data points are clustered into *k* groups, where each data point is closest to the mean of the group is belongs to. The algorithm is stochastic in that initial values are being picked at random, as illustrated in step 1, and depending on the initial conditions, there could be different results. Therefore, the algorithm should be repeated many times to find the optimal solution. The algorithm is recursive in that cluster means are repeated updated until the solution converges, as illustrated in step 4. [8]

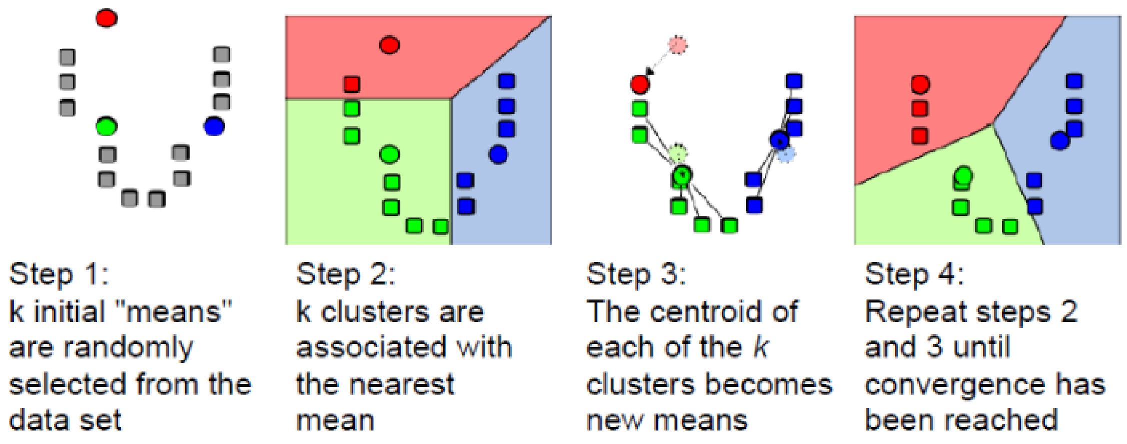

Step 1:
k initial "means" are randomly selected from the data set
Step 2:
k clusters are associated with the nearest mean
Step 3:
The centroid of each of the *k* clusters becomes new means
Step 4:
Repeat steps 2 and 3 until convergence has been reached

Figure 2 Steps in the K-Means algorithm

Closeness in our case is determined by the Euclidean distance between the points where all dimensions of the data point are accounted for, which is:

$$\sum_{x_j \in S_i} (x_j^2 - \mu_i^2)$$

where S_i is the set of all data points *x* in cluster *i,* and μ_i is the cluster center. Because distance is calculated this way, we require all data dimensions to be numerical and weighted according to their importance.

A. Data Cleaning and Transformation

Quantitative vs. Qualitative Data

Quantitative data such as disk counts and storage size can be directly fed into the K-Means algorithm as their values exactly convey cardinal information. However using storage size directly might cause some unreasonable skew in the statistics. The distribution of storage sizes is not linear, as customers either use a filer for storing big data with terabytes of information or for storing small data with gigabytes of information, with few customers using the filers for data sizes in between. To address this phenomenon, we take the logarithmic value of storage sizes, and use only their order of magnitude for K-Means to calculate the euclidean distance.

Qualitative data such as System Model, on the other hand, requires a more careful treatment because one cannot tell immediately whether one model is more similar to another just by their values.   For the System Model parameter, we propose two different solutions. One solution is to transform qualitative values into quantitative values based on their relative similarities. For example, Model FAS3210 and FAS3040 will be assigned close numerical values because they are functionally similar. The downside of this solution is that arithmetic operations would yield unreasonable results sometimes. For example, the average of two System Models might yield a numerical value that correspond to a third System Model, even though that third model might not be present at all. Our second solution address that problem by exploding the data parameter into many more data parameters, where each new data parameter is a binary value indicating whether the data point has a particular System Model. As a result, the original parameter is split into 12 different parameters.

Data Normalization and Weightings

The K-Means algorithm is designed to compress pairs of N-dimensional data into a one dimensional value represented as distance. In order to prevent any particular dimension to skew all N dimensions, normalization to the parameters is applied after the data transformation has taken place. All the parameters are normalized to one, through a process where each value is divided by the largest value in that parameter.

In addition to normalization, differential weights are also applied to the data parameters. This is to give more significance to data values that are important in terms of representing the data point, in our case, reflecting the characteristics of a filer machine. Weightings are also important to weigh down exploded parameters such as System Model, where it went from a one dimensional data to 12 dimensional data. This would significantly skew the results to reflect more importance in System Model if they were not scaled down accordingly. The actual weights to be used require subject area experts to determine.

B. Implementation of the Algorithm

To implement the K-Means approach for our analysis, we are using a standard C library for the core K-Means algorithm. We then developed a program that makes use of the K-Means algorithm specific for the Autosupport analysis. The basic steps of the program are illustrated in figure 3.

1. Parse out customer data, and apply appropriate data cleaning and transformation as described previously.
2. Run K-Means clustering on customer data to find out *k* cluster centers, as well as their associated cluster variance.
3. Parse out QA data, and apply appropriate data cleaning and transformation similar to step 1.

4. Associate each QA data point to its closest cluster center in terms of the Euclidean distance, and only if they are within two standard deviations of the cluster center

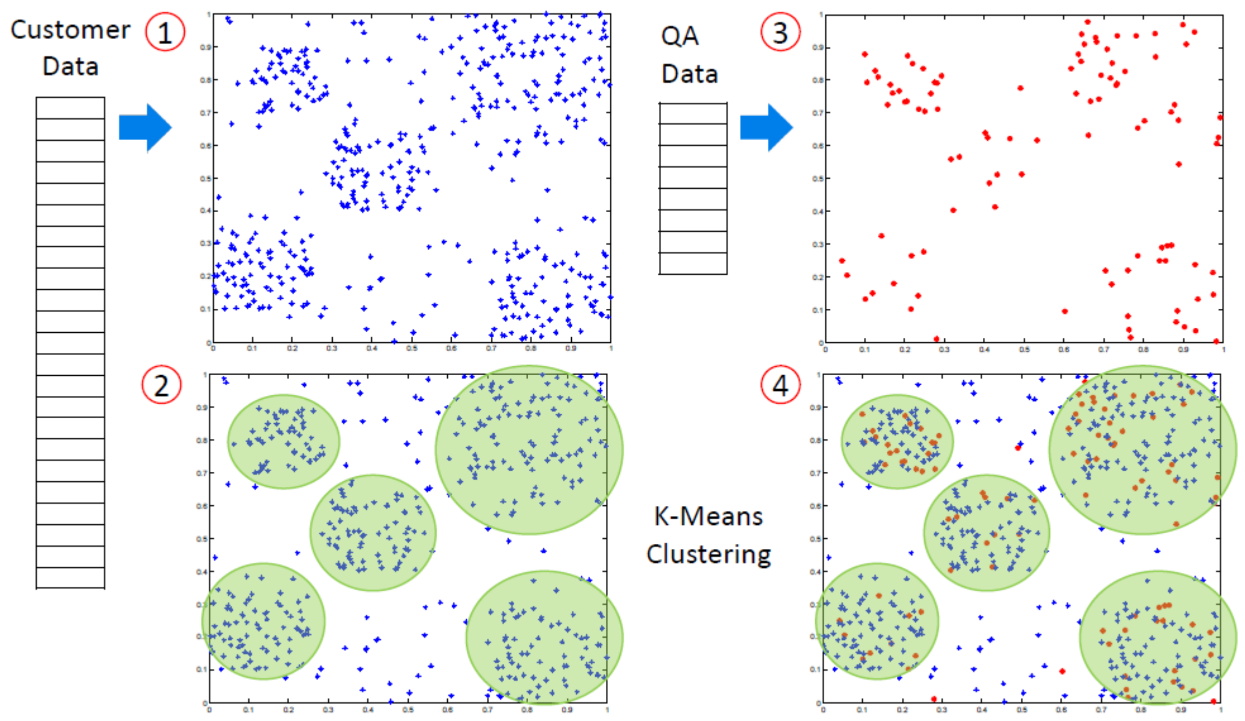


Figure 3 Summary of K-Means Implementation

The K-Means program implemented takes in a number of user specified parameter. Usage is detailed as the following:

```
gcc -o clusters *.c -lm

./clusters [in path] [out path] [columns to use] [column
weightings] [number of elements] [number of data dimensions]
[number of clusters] [number of repeats] [qa in path]
[number of qa]
```

We specify a number of repeats to run the K-Means algorithm to obtain the optimal solution, since the algorithm is stochastic and depending on the initial values, and the convergence solution is not necessarily unique.

The program will output four files for a particular *k*:

- output.csv – contains the cluster centers, the number of customer machines and the number of QA machines in each cluster, and the cluster variance associated with each cluster.
- output_id.txt – cluster ID associated with each customer machine.
- output_qa.csv – contains the QA machines that don't fit within two standard deviations of any cluster center.
- output_summary – contains a summary of the program run, including input parameters, and the total cluster variance.

VI. K-Means Algorithm Results

A. Common Customer Machine Configuration*s*

Because we have no prior knowledge as to what is the right number of clusters, *k*, to look for in the customer data set, we run the K-Means algorithm multiple times with different *k* value and use a parameter called percent variance explained to measure the clustering quality of a particular *k*. Percent variance explained is a variable that measures the cluster quality by calculating the relative variance of the data points to their cluster centers compared to the variance of the entire data set. It is calculated as,

$$\% \, variance \, explained = 1 - \frac{\sum_{i=1}^{k} \sigma_i^2}{\sigma_{total}}$$

Figure 4 shows the results of percent variance explained as a function of k. Of course, we will have a higher percent variance explained for higher *k*, as in the extreme case when k=number of data points, we will have perfect clustering for all data points. However, with higher number of k, we would lose semantic insight into the cluster itself. As we can see from the graph, at *k*=8, about 80% of the customer configurations are described by the cluster centers. For *k*>8, there is a diminishing return for the percent variance explained value. We therefore gave labels to the cluster centers to have more semantic insight into the grouping.

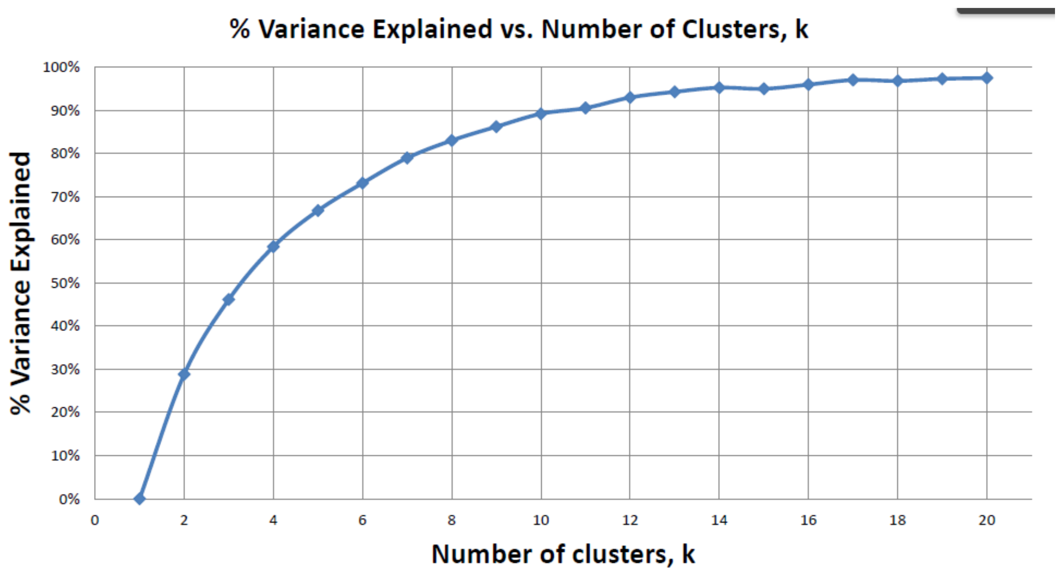


Figure 4 Percent Variance Explained

B. QA Machines Not Testing Any Customer Configurations

To find the extra QA machines that do not test any customer configurations, we list "loners" the QA machines that falls outside of two standard deviations of its closest cluster center during the K-Means calculation. As we obtain a list of QA machines from each K-Means run, the set of QA machines that repeated appear in the list of loner machines will have a higher probability of not belonging to any cluster of customer machines. These QA machines are not very useful in testing the real world customer configurations.

C. Customer Configurations Not Tested by Any QA Machine

To find out which group of customer configurations does have any QA machine testing it, we can increase the k value and look for clusters that have zero QA machines falling within two standard deviations of their cluster center. Such cluster does not seem to appear until k is increased beyond 100. These groups of customers would be prone to potential vulnerabilities as they do not have any similar QA machines testing them

VII. Conclusion

Given the results of our analysis, there are many recommendations we can make to the NetApp Company. However, we need to precisely determine which configuration parameters are the most important instead of relying solely on human intuition.

From results of both the direct matching analysis and K-Means clustering analysis, we can find out the vulnerable customer groups with little QA test coverage, and increase the QA effort on those groups. The results have also identified the QA machines which do not test any real world configurations, and these loner machines could either be reconfigured to test a customer configuration or be reduced in number for resource efficiency.

The eight common customer configurations that we have identified using the K-Means analysis can also prove useful to the development team. NetApp could monitor each customer group separately, and find out their specific needs in order to develop features and improvements targeted at each one. This would help with customer segmentation and increase overall customer satisfaction.

The marketing and sales team can also use the results of our analysis when approaching existing and prospective customers. With existing customers, our analysis will have demonstrated to them that their products are well tested internally at NetApp, and they will have higher confidence and satisfaction using NetApp's product. With prospective customers, the sales team can take the results of our analysis to show that the customer's product requirements have been well tested and have success stories with similar existing customers.

The actual adoption cycle of the above recommendations would require an iterative process. First, the QA team needs to make adjustments to their test machines according to the results of our analysis. Then, a carefully designed metric, such as development time, number of bugs reported over time, need to be tracked to provide feedback on the accuracy of our results. After that, we can learn from results of the metrics, and use better parameters and more appropriate weightings to do clustering analysis, which would in turn yield more accurate results to be implemented. The K-Means algorithm is quite scalable in terms of the number of parameters it can take in, as well as its potential to be implemented in distributed manner. Eventually, the QA test environment will be more representative of that out in the real world, and feature testing will be more relevant to customers.

In conclusion, the outcome of our analysis encourages a robust QA test environment that will have lasting impact extended to development and marketing and sales team. The methodology of our analysis could also be extended into other areas of business, where customer segmentation would be useful.

References

[1] S. e. a. Chakrabarti, "Data Mining Curriculum: A Proposal," 30 April 2006. [Online]. Available: http://www.sigkdd.org/curriculum.php.

[2] "Apache Hadoop," Apache Open Source, 19 March 2012. [Online]. Available: http://hadoop.apache.org.

[3] Y. e. a. Chen, "The Case for Evaluating MapReduce Performance Using Workload Suites," Singapore, 2011.

[4] J. Hambleton, Interviewee, *Quality Assurance Process at NetApp.* [Interview]. 28 April 2012.

[5] "NetApp Release Cycle," NetApp Inc., [Online]. Available: http://thebrewery-web.corp.netapp.com/Brewery/brewery/ReleaseProcess/LifeCycle/index.html. [Accessed 2 May 2012].

[6] NetApp Inc, [Online]. Available: http://wikid.netapp.com/w/Shared_Test_Beds/Process/Filer_ConfigTypes. [Accessed 2 May 2012].

[7] G. Fung, "A Comprehensive Overview of Basic Clustering Algorithms," p. 11, 2001.

[8] P.-N. M. S. V. K. Tan, Introduction to Data Mining, Addison Wesley, 2005.