Gentry Atkinson
CS7331, Spring 2019

Final Project Report: Using Parallelism to Speed Up Cluster Validation

1) Introduction:

Clustering is a well established and widely used tool in machine learning. It allows an analyst to associate unlabeled data into groups by means of some distance measure. But there are many decisions which need to be made correctly for the results of clustering to have any meaning. For example, an analyst will need to choose a distance measure, a clustering algorithm, and the parameters of the clustering algorithm. In order to validate that the correct decisions have been made the analyst must follow up with some method of cluster validation.

Cluster validation can be accomplished by external or by internal means. External validation makes comparisons to some ground truth provided by a domain expert. Although this is by far the most common means of cluster validation it really only validates the assumptions that an expert would make about a dataset rather than providing new insight. Aside from this shortcoming it is also frequently true that this groundtruth will not be known about freshly collected data. Using internal validation metrics allows analysts to sidestep both of these issues.

Internal validation indexes provide quantifiable measures of the validity of clusterings of data based solely on measuring the closeness of points which are group together by the clustering algorithm. There are many such indexes and choosing the correct on can be a burden in itself (unfortunately I've never found a validity index for validity indexes). This project will focus on the Dunn Index and the Davies-Boudlin Index because they are common, well established, and well supported but the reader should not take this as an indication that they are the best that the field has to offer.

The trouble with using internal validation on clusters is that they rely on expensive, each-to-each comparisons of data-points within the clusters. These calculations can often be as expensive as the task of clustering the data originally was. Since it would be a great utility to be able to rapidly and easily validate a set of clusters, then speeding up the process of validation will be a solid contribution to the field of unsupervised machine learning.

This project has undertaken to apply OpenMP to an implementation of the of the Dunn Index and of the Davies-Bouldin Index in order to measure the effect that parallelism can have on these two algorithms. To evaluate the performance of these implementations 3 datasets have been produced, and K-Means clustering with varying parameters have been generated. The validity of each clustering will be measured using varying degrees of parallelism on two different hardware platforms.

2) Tools:

2.1 The Dunn Index was first proposed by JC Dunn in the Journal of Cybernetics in 1974. It indicates the ratio between the minimum distance between any two clusters and the the maximum of the mean distance between points in any cluster. Because it divides by the max average distance,one poorly fit cluster can badly skew the index, so it is treated as the ‘worst case’ fittedness of any cluster in a set. **High values of the Dunn Index indicate better clusters**.

2.2 The Davies-Boudlin Index was first described by David Davies and Donald Bouldin in 1979 in IEEE Transactions on Pattern Analysis and Machine Intelligence. It represents a ratio of the distances of points to their own centroids, to the distance between centroids of clusters. It was proposed as a measure of the similarity of clusters and can be applied between clusters in much the same way that a distance measure can be applied between any two data points. Since dissimilar clusters indicate better clusters, **low values of the DB Index indicate better clusters.**

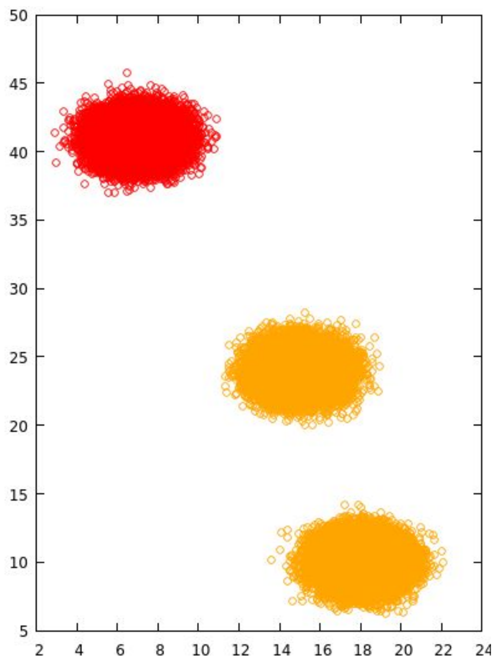
2.3 K-Means is a clustering algorithm which has been in common usage since 1967. It functions by guessing and then iteratively adjusting k cluster centers within some data. Although K-Means is not being tested or evaluated in this project, it will be used to generate the clusters that will then be evaluated by means of Dunn and Silhouettes.

2.4 OpenMP is an API which is used to add multithreading to C++ programs using a series of compiler directives. It was first released for Fortran in 1997 and was expanded to C++ in 2002. The use of #pragma directives within the code means that the parallelism can easily be turned on and off at compile time, essentially allowing for sequential and parallel executables to be produced from the same code.

3) Data:

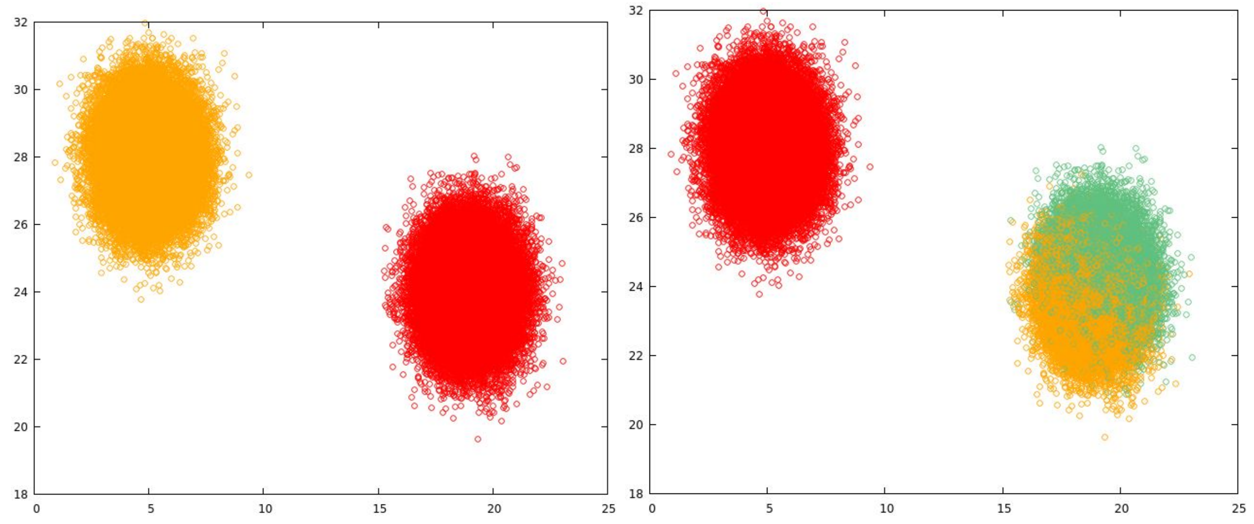
Three data sets have been generated for use in this project and from each of them a set of possible clusterings have been generated with the K-Means clustering algorithm.

3.1 Set 1 consists of 100,000 two dimensional data-points distributed normally around 3 randomly chosen centers. Clustering were generated using k=2, and k=3. Because the data has been generated around three centers we hope to see our validity indices showing that k=3 has generated the better result. The low dimensionality of this dataset will make it much easier to visualize than the other two.



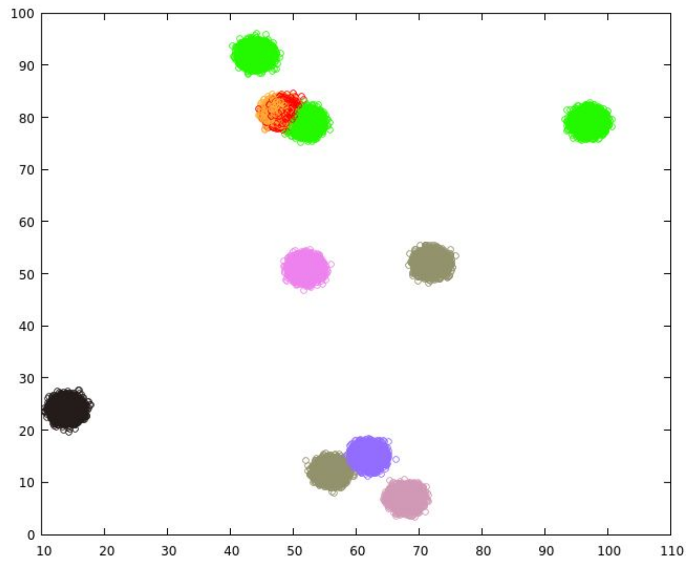
Data Set 1 clustered with k=2 and k=3.

3.2 Set 2 consists of 100,000 ten dimensional data-points distributed normally around 2 randomly chosen centers. Again 2-means and 3-means clustering was applied to this set although with this set we will hope to see k=2 producing the better results. Because of the higher dimensionality of this data, we will have to rely on various projections of the data in order to give us some understanding of its structure.

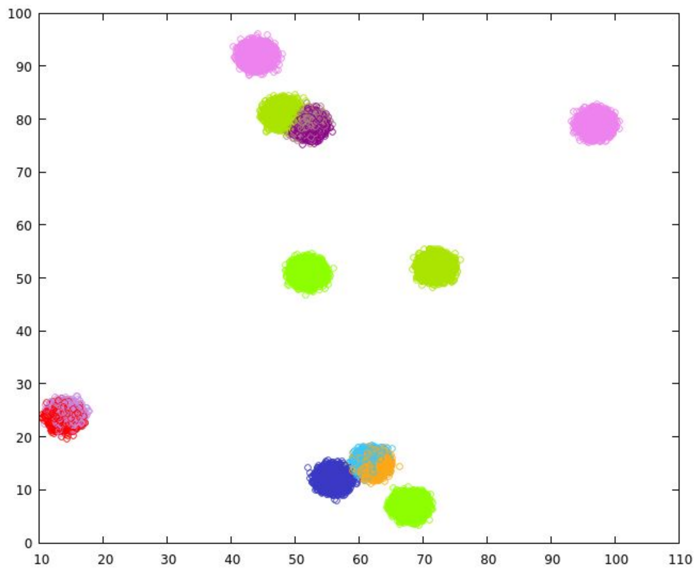


Data Set 2 clustered with k=2 and k=3 and projected onto dimensions 1 and 2.

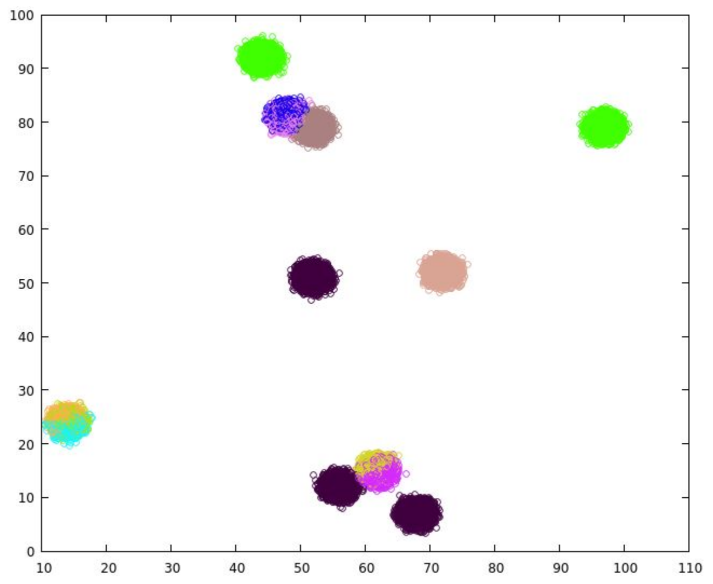
3.3 Set 3 consists of 100,000 ten dimensional data-points distributed normally around 10 randomly chosen centers. K-Means clustering was applied with k=8, 10, and 12. Because of the possibility that some clusters will overlap, we cannot be certain from the outset that 10-Means will produce the most valid result. Rather we will have to rely on our validity measures to determine which set of clusters best fit the data in much the same way that an analyst would. Visualization of this set is substantially more difficult than the other two and clusterings that appear tight or loose in one projection may be the opposite in another. This is often also true with real world data and adds to the applicability of the results derived from this project.



Set 3 clustered with k=8 and projected onto dimensions 1, 2.



Set 3 clustered with k=8 and projected onto dimensions 1, 2.



Set 3 clustered with k=12 and projected onto dimensions 1, 2.

4) Experimental Setup:

Datasets 1 and 2 were each used to produce two experimental clusterings. Dataset 3 has contributed 3 experimental clustrings. This gives a total of 7 clusterings to run the experimental algorithms. Implementations of the Dunn Index and the Davies-Boudlin Index were written in C++ with with OpenMP library. In each implementation the number of threads to run can be passed into the program at runtime. The implementations have also had timers written in to clock the execution of the algorithms using the Chrono library. This was done rather than using an external tool such as perf-stat because both implementations incorporate a large file reading. Since this IO process is unrelated to the algorithms being tested, timing it would not contribute anything meaningful to this project. Finally, each implementation was compiled using gcc at the lowest level of optimization (-O0). This was done to isolate the single variable being tested by this project (parallelism) rather than measuring the optimality of a commonly available compiler. It is worth noting that some runs were done with the compiler at a higher level of optimization (-O2) during testing and that it dramatically improved the performance of the executables but that data was not collected or analyzed.

4.1 Test Machine 1 is a home laptop with the following hardware information generated by lscpu:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):          8
```

```
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):       1
NUMA node(s):          1
Vendor ID:             GenuineIntel
```

4.2 Test Machine 2 is the Zeus server provided by the CS department of Texas State University. The following hardware information is generated by lscpu:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             4
NUMA node(s):          1
Vendor ID:             GenuineIntel
```

5) Results:

The first measure of interest to us is the values of the indexes computed by our experimental implementations. If these do not reflect the structure of the data generated then our implementations are not functioning well and any further results will not have any meaning. Once it is established that validity measures are functioning as expected than we can further understand the effect that parallelism on the performance of our indexes.

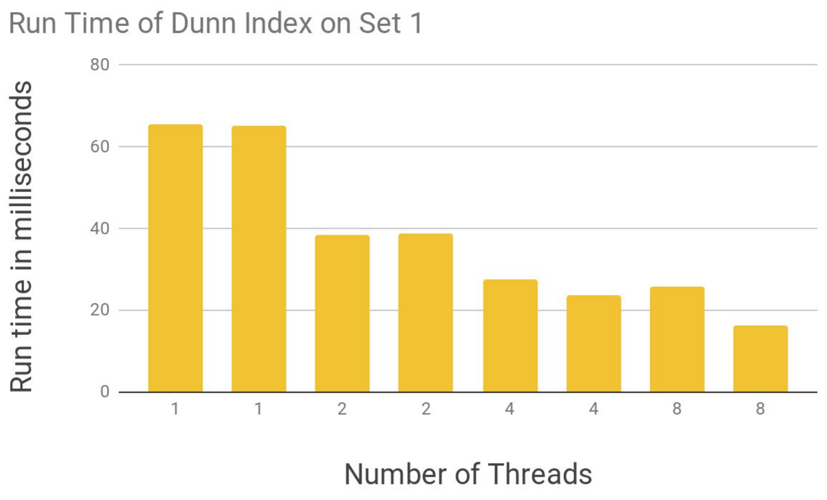
Set 1:	**2 clusters**	**3 clusters**
Dunn Index	2.287	2.988
DB Index	0.164	0.084

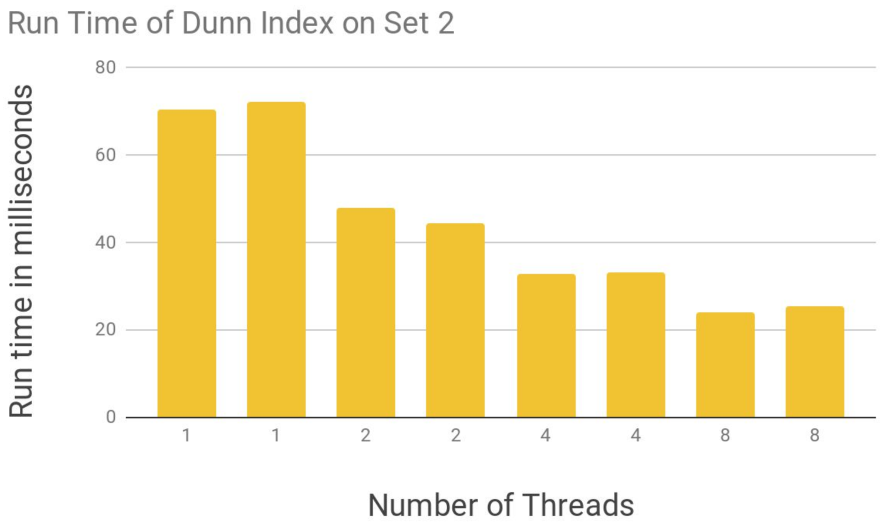
Set 2:	**2 clusters**	**3 clusters**
Dunn Index	3.668	0.239
DB Index	0.124	1.308

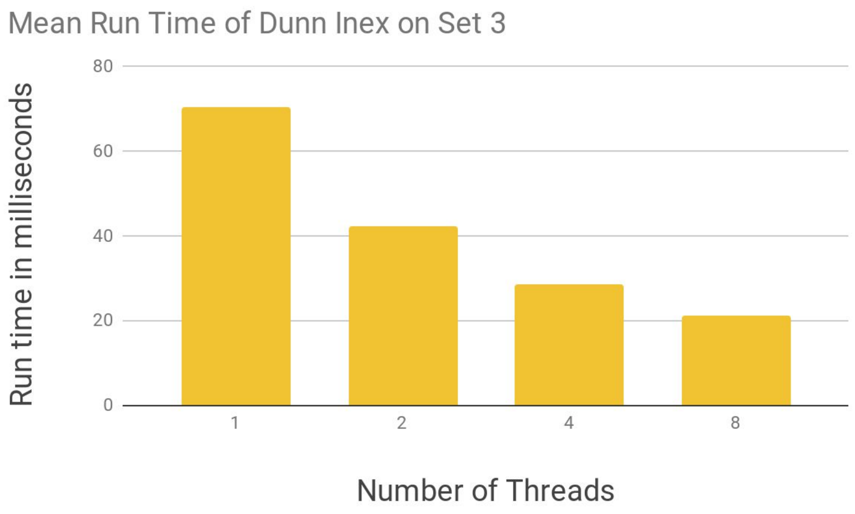
Set 3:	**8 clusters**	**10 clusters**	**12 clusters**
Dunn Index	0.021	0.025	0.027
DB Index	0.674	1.5	1.327

5.1 Validity Measures gathered on all 3 data-sets are promising. Recall from Section 2 that the Dunn Index is consider to be a better clustering with higher values, while the Davies-Bouldin index is considered to be a better clustering with lower values. Set 1 exhibits a higher Dunn Index and lower DB Index with 3-Means, matching our expected result. Set 2 exhibits the higher Dunn Index and lower DB Index with 2-Means, which is again inline with what we would expect to see on this set. Set 3 produced three clusters of roughly equal Dunn Indices but with 8-Means showing a much better (lower) DB Index. This makes sense in light of the fact that some cluster may overlap and "merge" in cases with a greater number of centers, as was mentioned in Section 3.

5.2 Machine 1 Dunn Performance Measures:

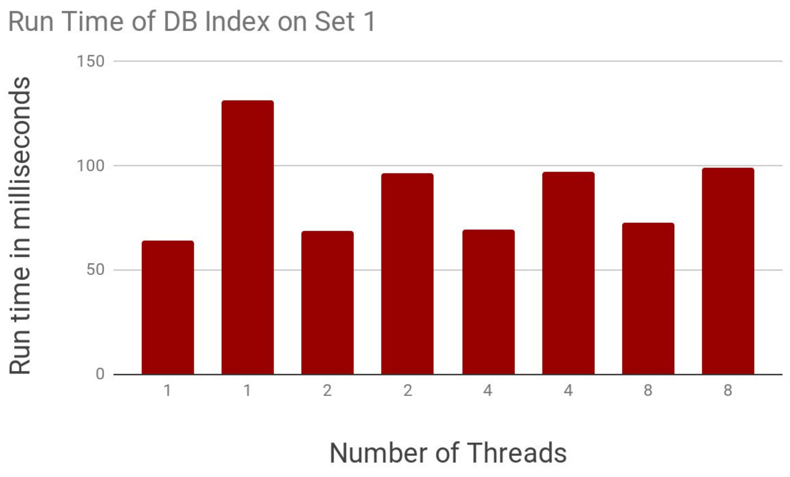


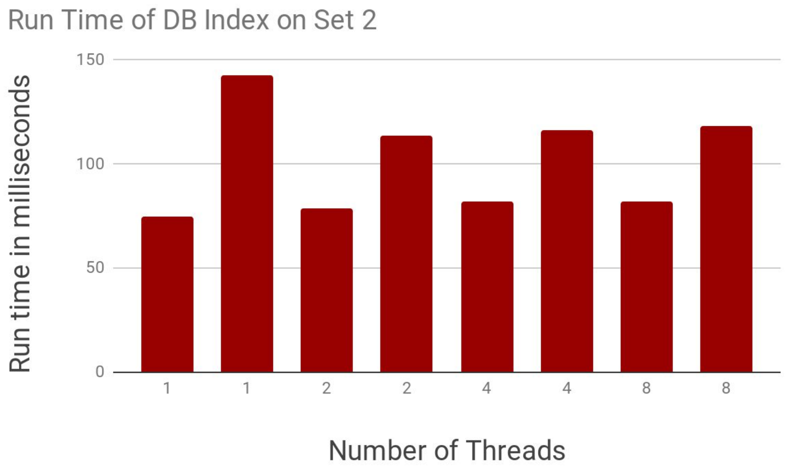


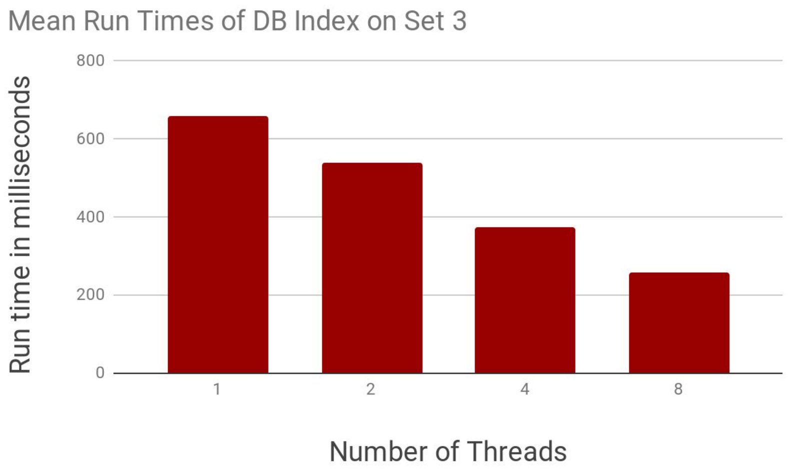


The Dunn Index improved its performance across all three datasets. The data for set 3 is presented as an arithmetic mean of the runs for k=8, k=10, and k=12 to make the chart more presentable. It also demonstrates a very smooth variance as the the number of threads for the run were increased, with 1 thread giving a runtime of 70.571 milliseconds down to 21.085 milliseconds on the 8 thread run.

5.3 Machine 1 DB Index Performance Measures:

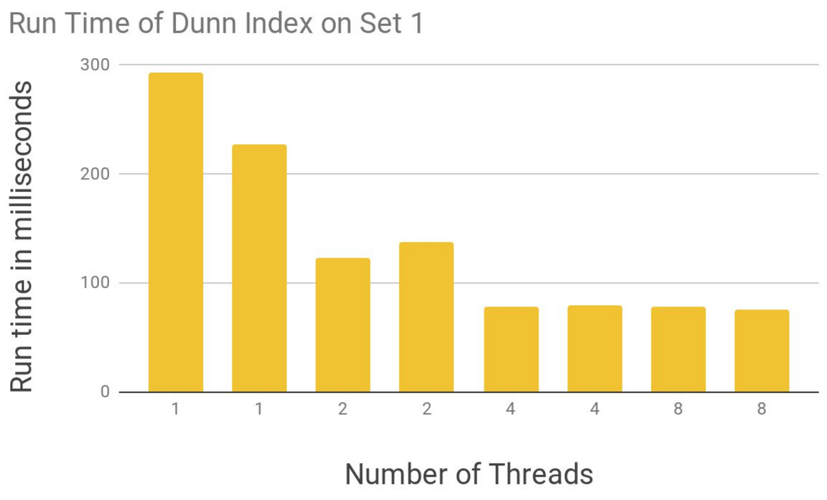


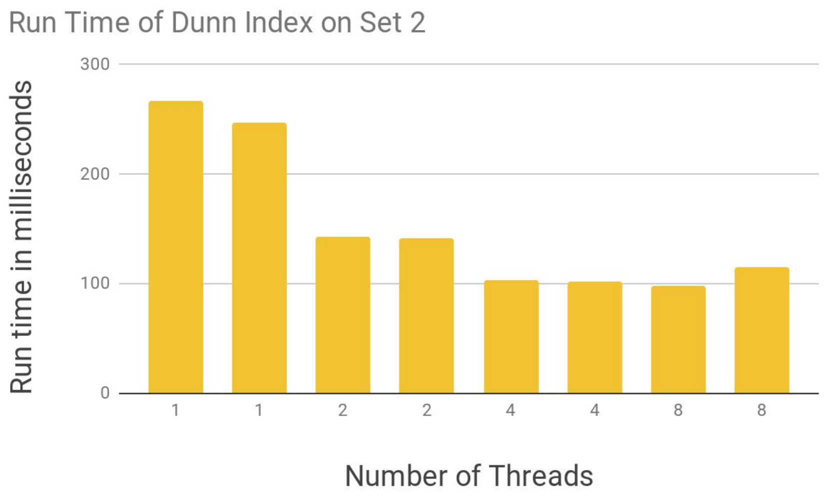


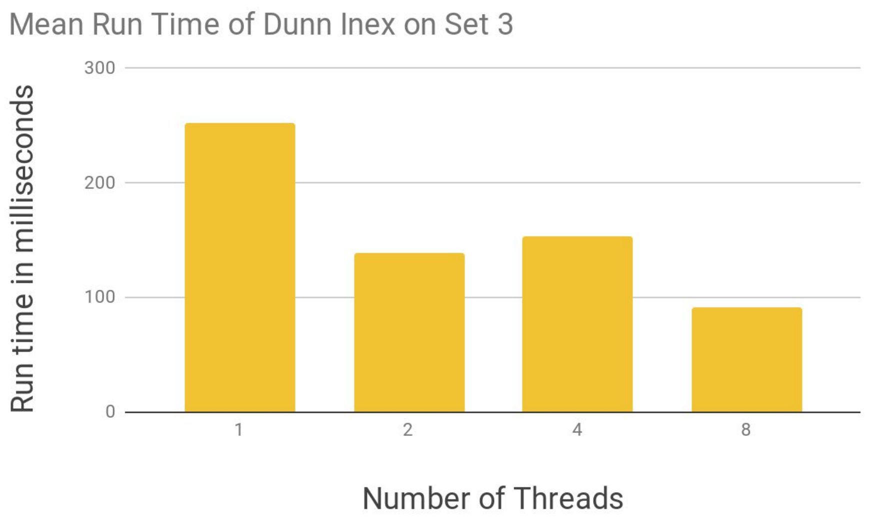


The Davies-Boudlin index produces figures with a less distinct improvement. Analysis of the code showed that an outer loop in the code iterates over the number of clusters in the set rather than iterating across all points. It is therefore reasonable that cluster sets with a low number of centers will not benefit much from the performance improvements offered by parallelism. However, the clusterings of Set 3, which is again charted using means of runtimes on k=8, k=10, and k=12, showed significant improvement as the number of threads increased. The single thread runs terminated in an average of 657.9 milliseconds while the 8 threads runs terminated in 257.8 milliseconds.

5.4 Machine 2 Dunn Index Performance Measures:

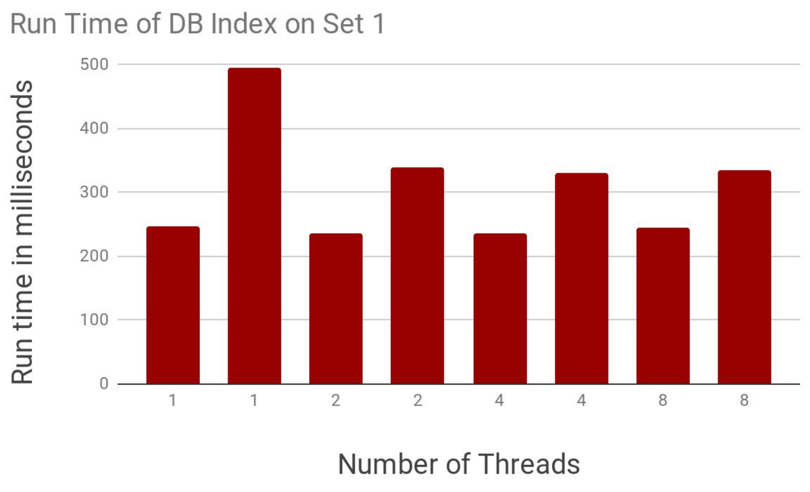


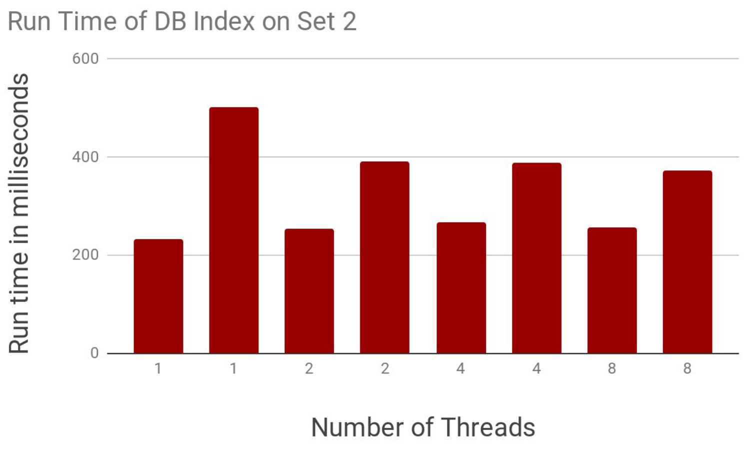


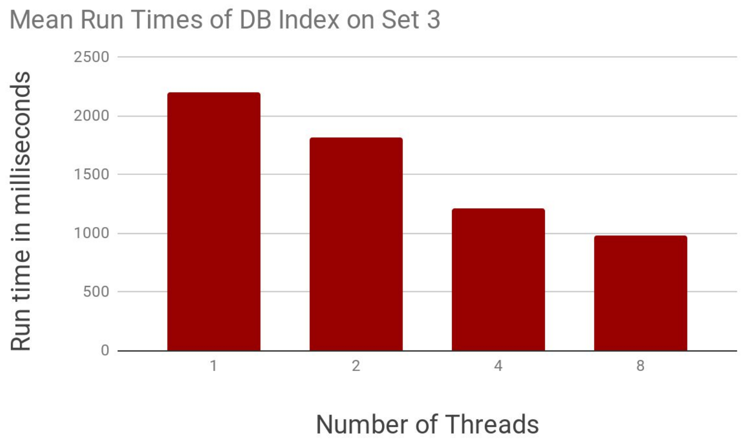


The Dunn Index implementation running on Machine 2 shows similar results to Machine 1 up to a point. However the runs with 8 threads allocated show no performance on Sets 1 and 2. This is consistent with the information provided in Section 4 which indicated that that Machine 2 is a 4-core machine, while Machine 1 is an 8-core. Set 3 also exhibits and unusual artifact. Of the 3 runs with 4 threads one terminated in 88ms, one in 101ms, and one in 271 ms. I have found no good reason that one run would take 3 times as long as its fellows and can only speculate that Zeus was busy with some other process for a short time.

5.5 Machine 2 DB Index Performance Measures:







As was observed on Machine 1, Sets 1 and 2 demonstrated little if any performance improvement as threads were added. We can also set that Set 3 benefits from the addition of threads, going from a 2206 ms run with 1 thread down to a 986 ms run with 8 threads.

5.6 Analysis of Results:

It's reasonable to say broadly that the introduction of parallelism improved the performance of both algorithms, with considerations. In this case as in most a developer using these tools will need to understand both the problem and the hardware that he or she is working on in order to earn the greatest result. The failure of the DB Index to show profitability as threads were added in Sets 1 and 2 show the importance of understanding the problem. Similarly the inability of Machine 2 to profit from the addition of threads beyond the 4th show the importance of understanding the hardware which will be processing a problem.

Machine 2 generally shows much greater run times than Machine 1, but also generally shows a similar slope between runtimes with increasing number of threads. The varying number of processors accounts for the difference in the highest thread runs but outside of those cases the performance improvements are quite similar. This shows that the two algorithms achieve very similar execution in terms of other metrics like cache and memory locality. Any difference in the two machine in this regard would exhibit itself in some variation of the performance improvements in runs with 1, 2, and 4 threads allocated.

6) Conclusions:

Overall this project has shown positive results. This work has many future applications. One promising instance is hierarchical clustering algorithms. Hierarchical clustering algorithms cluster a dataset into a low number of clusters and then re-divide each cluster in to sub-clusters. The user receives a tree of the possible sets of clusters. Effective cluster validation will allow the level of the tree which offers the best fit and improving the performance of those measures will offer an exponential return as the branching sets and subsets are all validated. Another possible application would be in non-deterministic clustering algorithms, such as K-Means. Many clustering algorithms cannot guarantee that the same set of clusters will be returned by sequential runs with the same parameters. A fast and effective means of validating clusters will allow the best result to be quickly chosen from several runs of an algorithm.

But this promise should be tempered with caution. A complete understanding of the various factors of a system is necessary to achieve the greatest result from adding parallelism to these systems. Some problems will exhibit a strongly sequential execution and will not benefit from the addition of parallelized code. Similarly the hardware must always dictate the degree of parallelism that is permitted in a system. But with the right paring, these systems have shown that parallel execution is the right choice for cluster validation.

7) Acknowledgements:

I would like to thank openmp.org for their free and easy provision of the OpenMP library for c++. This project wouldn’t have been possible without their generous spirits.