International Journal of Scientific and Technical Advancements

ISSN: 2454-1532

An Efficient Approach for Clustering High Dimensional Data

R. Suganya[1], S. Vydehi[2]

[1, 2]Department of Computer Science, Dr. SNS Rajalakshmi College of Arts and Science, Coimbatore, Tamil Nadu, India

Email address: [1]rsuganya.2292@gmail.com, [2]vydehiela@gmail.com

***Abstract*—**Big data analytics allows a small number of users to burn a large amount of money very fast. The problem is exacerbated by the exploratory nature of big data analytics where queries are iteratively refined, including the submission of many erroneous (e.g., big streaming data cluster). In existing systems, clustering must complete after downloading, often after several hours of expensive compute time are used for clustering. This project had shown that it is both progressive fetching and clustering to support incremental query interactions for data analysts. High Dimensional (HD) clustering has been successfully used in a lot of clustering problems. However, most of the applications deal with static data. This project considers how to apply HD in incremental clustering problems. Clustering data by identifying a subset of representative examples is important for detecting patterns in data and in processing sensory signals. Such "exemplars" can be found by randomly choosing an initial subset of data points as exemplars and then iteratively refining it, but this works well only if that initial choice is close to a good solution. This thesis describes a method called "Big Data Clustering using k-Mediods BAT Algorithm" KMBAT, that simultaneously considers all data points as potential exemplars, exchanging real-valued messages between data points until a high-quality set of exemplars and corresponding clusters gradually emerges. KMBAT takes as input a set of pairwise similarities between data points and finds clusters on the basis of maximizing the total similarity between data points and their exemplars. Similarity can be simply defined as negative squared Euclidean distance for compatibility with other algorithms, or it can incorporate richer domain-specific models (e.g., translation-invariant distances for comparing images). KMBAT's computational and memory requirements scale linearly with the number of similarities input; for non-sparse problems where all possible similarities are computed, these requirements scale quadratic ally with the number of data points from big data which is streamed. KMBAT is demonstrated on FACEBOOK social network user profile data, which is stored in a big data HDInsight server and cluster with KMBAT which finds better clustering solutions than other methods in less time.

***Keywords*—**Clustering; BAT algorithm; k-medoids; HADOOP.

I. INTRODUCTION

In Data Warehousing and Data Mining, storing the data, analyzing the data, processing the data and managing the data cannot be done in parallel. It cannot handle both structured and unstructured data at a time. Data Warehousing and Data mining spend 95% of the time on gathering and retrieving the data and only 5% of the time is spend for analyzing the data. But in real time scenario, we are in a situation to analyze each and every data. We are generating data faster than ever, so the need for Bigdata emerged. In Big data 70% of the time is spend on gathering and retrieving the data and remaining 30% of the time is spend on analyzing the data. For example, twitter have to process 340 million messages per day where as Amazon S3 storage should add more than one billion objects a day, in case of Facebook it should handle 2.7 billion "comments" and "likes" generated per day by its users. All the above is possible, with the help of Big Data. Big data is capable of handling large datasets at a time. It can perform data storage, data analysis, and data processing and data management techniques in parallel. Big data is a popular term used to describe the exponential growth and availability of data, both structured and unstructured. Big data spends 70% of the time on gathering and retrieving the data and remaining 30% of the time is spend on analyzing the data. Big data can process even several petabytes of data in seconds. Big data analytics will be most useful for hospital management and government sectors especially in climate condition monitoring. Bigdata analytics allows a small number of users to burn a large amount of money very fast. The problem is exacerbated by the exploratory nature of big data analytics where queries are iteratively refined, including the submission of many erroneous (e.g., bad query parameters) and off-target queries.

A. *Clustering Big Data*

Cluster analysis seeks to discover groups, or clusters, of similar objects. The objects are usually represented as a vector of measurements, or a point in multidimensional space. The similarity between objects is often determined using distance measures over the various dimensions in the dataset. Technology advances have made data collection easier and faster, resulting in larger, more complex datasets with many objects and dimensions. As the datasets become larger and more varied, adaptations to existing algorithms are required to maintain cluster quality and speed. Traditional clustering algorithms consider all of the dimensions of an input dataset in an attempt to learn as much as possible about each object described. In high dimensional data, however, many of the dimensions are often irrelevant. These irrelevant dimensions can confuse clustering algorithms by hiding clusters in noisy data. In very high dimensions it is common for all of the objects in a dataset to be nearly equidistant from each other, completely masking the clusters. Feature selection methods have been employed somewhat successfully to improve cluster quality. These algorithms find a subset of dimensions on which to perform clustering by removing ir-relevant and redundant dimensions. Unlike feature selection methods which examine the dataset as a whole, subspace clustering algorithms localize their search and are able to uncover clusters that exist in multiple, possibly overlapping subspaces.

B. Motivation

With the rising of data sharing websites, such as Facebook and Flickr, there is a dramatic growth in the number of data. For example, Facebook reports about 6 billion new photos every month and 72 hours of video are uploaded to YouTube every minute. One of major data mining tasks is to unsupervised categorize the large-scale data. In the Project Description chapter the clear problem analysis is made, which helps to understand the problem definition and the overview of the research work. Also the process flow of the research, the sequential process and finally the estimated time limit of the research completion are also analyzed.

II. Literature Support

Data mining environment produces a large amount of data that need to be analyzed; patterns have to be extracted from that to gain knowledge. It has become difficult to process, manage and analyze patterns using traditional databases and architectures. This presents a review of various algorithms necessary for handling such large data set. To extract patterns and classify data with high similar traits, Data Mining approaches such as Genetic algorithm, neural networks, support vector Machines, association algorithm, clustering algorithm, cluster analysis, were used. Big Data architecture typically consists of three segments:

- Storage system
- Handling and
- Analysis

Big Data typically differ from data warehouse in architecture; it follows a distributed approach whereas a data warehouse follows a centralized one. The Data Mining termed Knowledge; its architecture was laid describing extracting knowledge from large data. Data was analyzed using software Hive and Hadoop. For the analysis of data with different format cloud structure was laid.

In 2004, Map Reduce was proposed by Google, it is an object-oriented programming model to deal with the large data, primarily used for processing internet data. The Map Reduce technology includes two basic operation conceptions:

- Map (Mapping) and
- Reduce (Simplication).

The Map technology mainly processes a group of input data record and distributes data to several servers and operation systems. Its means of processing data is a strategy based on the key/value. The Reduce technology mainly occupies itself in summarizing and processing the result after processing the above key/value. Map Reduce is designed for mass composed of low-end computer cluster, its excellent scalability has been fully verified in industry. Map Reduce has low requirement to hardware. Map Reduce can store data in any format; can achieve a variety of complex data processing function. Analysis based on the Map Reduce platform, without the need of complex data preprocessing and writing in the database process.

With the availability of large-scale computing platforms for high-fidelity design and simulations, and instrumentation for gathering scientific as well as business data, increased emphasis is being placed on efficient techniques for analyzing large and extremely high-dimensional data sets. Analysis of high-dimensional data typically takes the form of extracting correlations between data items, discovering meaningful information in data, clustering data items, and finding efficient representations for clustered data, classification, and event association. Since the volume (and dimensionality) of data is typically large, the emphasis of new algorithms must be on efficiency and scalability to large data sets. Analysis of continuous attribute data generally takes the form of Eigen value/ singular value problems (PCA/rank reduction), clustering, least squares problems, etc. Analysis of discrete data sets, however, generally leads to NP complete/hard problems, especially when physically interpretable results in discrete spaces are desired.

Compression of discrete data is a particularly challenging problem when compressed data is required to directly convey the underlying patterns in the data. Conventional techniques such as singular value decomposition (SVD), frequency transforms such as discrete cosine transforms (DCT) and wavelets, and others do not apply here because the compressed data (orthogonalzed vectors or frequency coefficients) are not directly interpretable as signals in noisy data. Techniques for clustering do not generalize easily to extremely high dimensions (104 or more) while yielding error-bounded cluster centroids. Unfortunately, the runtimes of all these methods are unacceptably large when scaled to millions of records of very high dimension.

A. Full-Data processing: Data is stored or cached in (distributed) main memory, and uses efficient organizations such as columnar formats, in order to allow queries over the entire data to complete in a very short time. Examples of such systems include Dremel [6] and PowerDrill [7].

B. Progressive processing: An alternative paradigm that can better fit a low-cost iterative querying paradigm is progressive processing, where the system produces early results based on partially processed data, and progressively refines these results as more data is received; until all the data is read, at which point the final result is produced. Progressive processing allows users to get early results using significantly fewer resources, and potentially end (or reissue) computations early once sufficient accuracy – or an early indication of query incorrectness – is observed. Several systems fall under the umbrella of progressive analytics, including the CONTROL project [3], the DBO system [5], and Map-Reduce-Online [6]. In this paper cluster the incremental data using kmediod bat algorithm. Incremental analysis is an alternative to other techniques that are more familiar, but have disadvantages compared to our method. In this section, we first discuss these techniques in order to motivate incremental data analysis. We then discuss techniques for visualizing uncertainty, which we adapt for our clustering.

III. PROPOSED FRAMEWORK

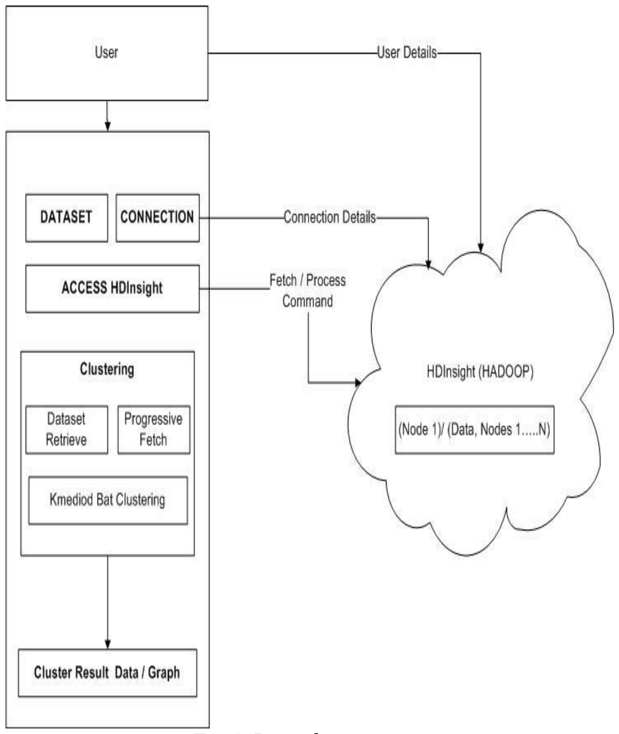

Fig. 1. Integral component.

A. *Social Connection*

Social Intranets are the cornerstone of company-wide collaboration. This system Dashboard is a customizable start page, which displays the information that matters to you, all in one place. Share status updates, discover popular content and connect with the people who can help you get your job done. This system has dashboard is an students personal home page linking to everything they need, as well as providing visibility into changes across the college for a quick determination into what needs attention and involvement.

(i) Communications platform

Social Enterprise dashboards come with administrative control that can be locked in place and used for companywide communications such as announcements and events. Enabling an efficient line of communication within the enterprise helps students to understand department initiatives and positioning and leads to a more cohesive and empowered communication that understands the college vision and their role in executing to that vision.

(ii)Social profile

Social Enterprise is accessible any time from multiple platforms such as the web, mobile devices or your desktop client. No matter where you are, you'll always be able to access your critical college content, and connect and share with your group. Student specifies their social profile ids. This system stores the id and mines for the social usage information.

B. *API Interactions and Notification*

The Graph API is the core of Facebook Platform, enabling developers to read from and write data into Facebook. The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend relationships, shared content, and photo tags). The Graph API is the primary way to get data in and out of Facebook's social graph. It's a low-level HTTP-based API that you can use to query data, post new stories, upload photos and a variety of other tasks that an app might need to do.

The Graph API is named after the idea of a 'social graph' - a representation of the information on Facebook composed of:

- nodes (basically "things" such as a User, a Photo, a Page, a Comment)
- edges (the connections between those "things", such as a Page's Photos, or a Photo's Comments)
- fields (info about those "things", such as the birthday of a User, or the name of a Page).

The Graph API is HTTP based, so works with any language that has an HTTP library, such as cURL, urllib. We'll explain a bit more about what you can do with this in the section below, but it means you can also use the Graph API directly in your browser once the student usage detail is mined to that profile the notification is sent.

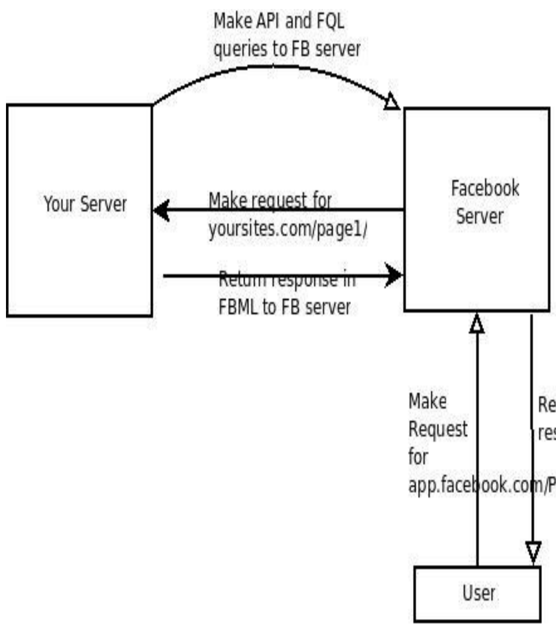
Make API and FQL queries to FB server
Your Server
Make request for yoursites.com/page1/
Return response in FBML to FB server
Facebook Server
Make Request for app.facebook.com/Page1/
Return response
User

Fig. 2. Graph API.

C. *Hadoop Connection*

Hadoop is implemented as a set of interrelated project components. The core components are MapReduce, which handles job execution, and a storage layer, typically implemented as the Hadoop Distributed File System (HDFS) this project uses Windows Azure HDInsight for HDFS. In HDInsight Hadoop components are implemented across a series of servers referred to as Blobs. These blobs are where data are stored and processed. A name blob server keeps track of the data blobs in the environment, which data are stored on which blob, and presents that data blob as a singular entity. This singular representation is referred to as a cluster. Once the process is complete, you have a working HDInsight cluster. The cluster consists of data nodes, a name node, and an associated storage account delivered through the Azure Storage service. The portal will show you the HDInsight cluster as soon as provisioning is completed but to see the storage account, you may need click the HOME link at the top of the portal and then click PORTAL from the Azure homepage to return to the portal page. The storage account should now be visible.

(i) Loading data to hadoop clusters

Hadoop presents a REST interface on HTTP port 50070. And while you could program data loads directly against that interface, the .NET SDK makes available a Web HDFS client to simplify the process. To make use of the Web HDFS client, you must have knowledge of which storage system is used within the cluster to which you are loading data. By default, the Web HDFS client assumes the target cluster employs HDFS. In this post, we will focus on the use of the Web HDFS client against our local desktop development cluster, which makes use of HDFS.

CONNECT TO HD INSIGHT

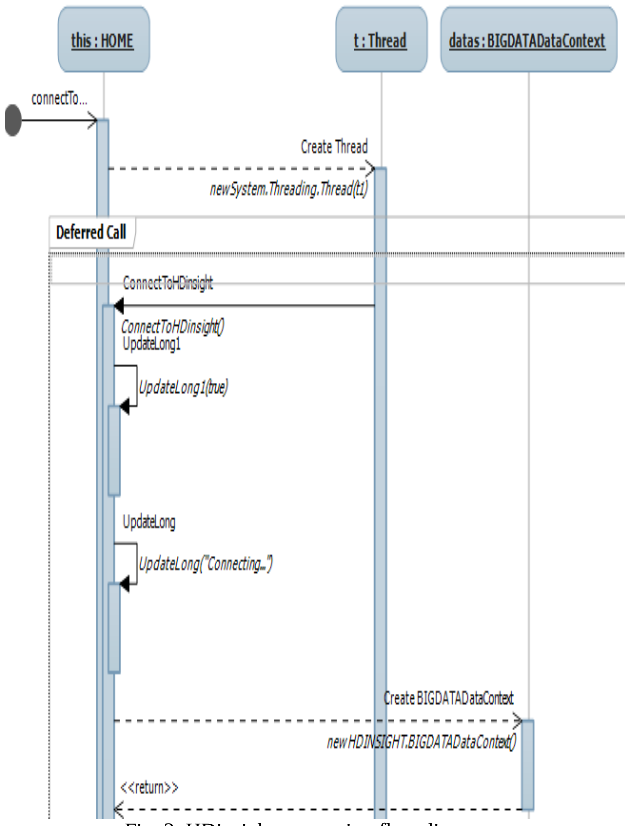

Fig. 3. HDinsight connection flow diagram.

DATASET LOADING SEQUENCE

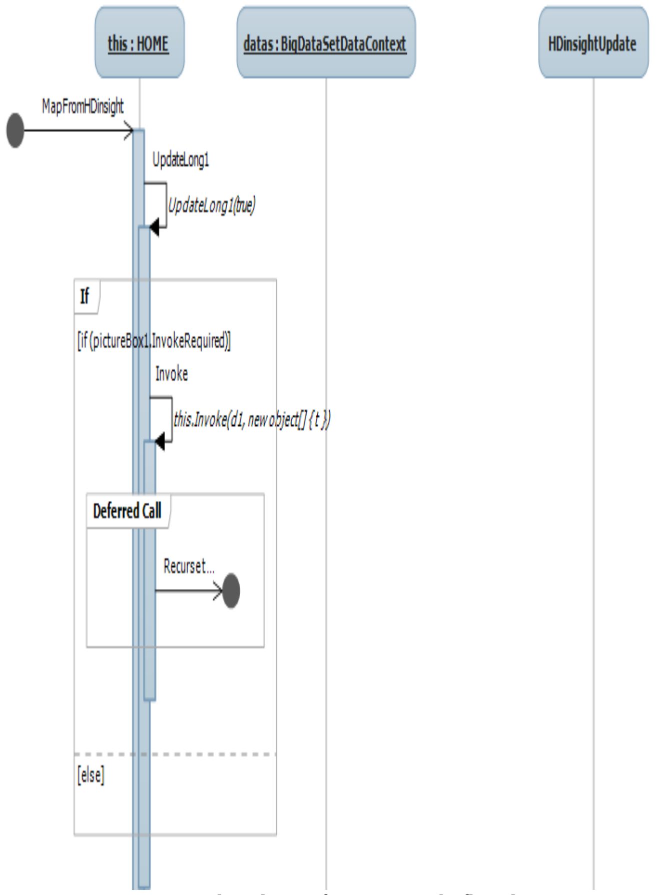

Fig. 4. Loading dataset from HDinsight flow diagram.

D. Progressive Fetching

This query should look familiar to any developer with experience using LINQ. Because this project uses LINQ as its query language, this query looks just like a LINQ to SQL query hitting a database or an in-memory filtering of an IList a class in .net framework. As events arrive from the input adapter, their payloads are inspected, and if the value of the Value property is greater than 0.5, they're passed to the output adapter where they're printed to the console.

When the application runs, notice that events continually arrive in the output. This is effectively a push model. This project computes new output events from inputs as they arrive, rather than a pull model like a database where the application must periodically poll the data source to see if new data has arrived. This fits nicely with the support of IObservable available in the Microsoft .NET Framework 4.

Having a push model for continuous data instead of polling is nice, but the real power of progressive fetching becomes apparent when querying over properties relating to time. As events arrive through the input adapter, they're given a timestamp. This timestamp may come from the data source itself (suppose the events represent historical data with an explicit column storing the time) or can be set to the time the event arrived. Time is, in effect, first class in the querying language of this project.

Queries often look like standard database queries with a time qualifier stuck on the end, such as “every five seconds” or “every three seconds over a five-second span.” For example, here’s a simple query that finds the average of the Value property every five seconds:

```
var aggregated = from i in inputStream
.TumblingWindow(TimeSpan.FromSeconds(5),
HoppingWindowOutputPolicy.ClipToWindowEnd)
select new { Avg = i.Avg(p => p.Value)};
```

(i)Windows of data

Because the concept of time is a fundamental necessity to complex event-processing systems, it’s important to have a simple way to work with the time component of query logic in the system. This project uses the concept of windows to represent groupings by time. The previous query uses a tumbling window. When the application runs, the query will generate a single output event every five seconds (the size of the window). The output event represents the average over the last five seconds. Just like in LINQ to SQL or LINQ to Objects, aggregation methods like Sum and Average can roll up events grouped by time into single values, or Select can be used to project the output into a different format.

Tumbling windows are just a special case of another window type: the hopping window. Hopping windows have a size, too, but they also have a hop size that isn’t equal to their window size. This means hopping windows can overlap each other.

For example, a hopping window with a window size of five seconds and a hop size of three seconds will produce output every three seconds (the hop size), giving you the average over the last five seconds (the window size). It hops forward three seconds at a time and is five seconds long. figure 6 shows an event stream grouped into tumbling and hopping windows.

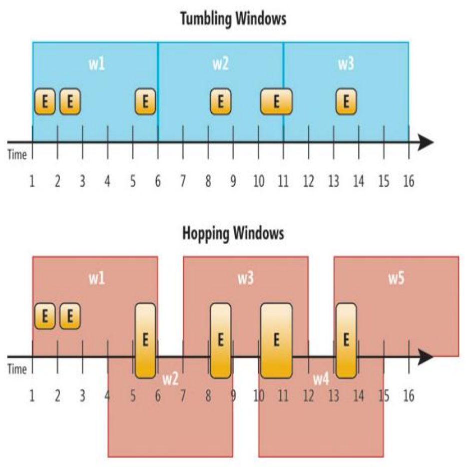

Fig. 6. Tumbling and Hopping Windows.

Notice that the tumbling windows do not overlap, but the hopping windows can if the hop size is smaller than the window size. If the windows overlap, an event may end up in more than one, like the third event, which is in both windows 1 and window 2. Edge events (that have duration) may also overlap window boundaries and end up in more than one window, like the second-to-last event in the tumbling window. Another common window type is the count window. Count windows contain a specific number of events rather than events at a particular point or duration of time. A query to find the average of the last three events that arrived would use a count window. One current limitation of count windows is that the built-in aggregation methods like Sum and Average are not supported. Instead, you must create a user-defined aggregate. This simple process is explained later in the article.

The final window type is the snapshot window. Snapshot windows are easiest to understand in the context of edge events. Every time an event begins or ends, the current window is completed and a new one starts. Figure 7 shows how edge events are grouped into snapshot windows. Notice how every event boundary triggers a window boundary. E1 begins and so does w1. When E2 begins, w1 is completed and w2 begins. The next edge is E1 ending, which completes w2 and starts w3. The result is three windows: w1 containing E1, w2 containing E1 and E2, and w3 containing E3. Once the events are grouped into the windows, they’re stretched so that it appears that the event begins and ends when the window does. In this way progressive way works.

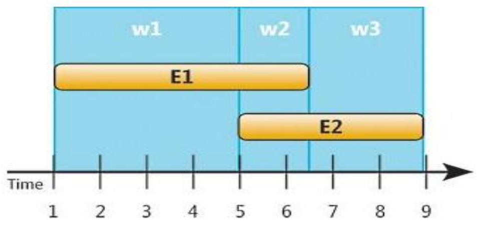

Fig. 7. Snapshot window.

E. K-Medoids Clustering Algorithm

K-Medoids is the most popular clustering algorithm in which a data set of n object is clustered with k number of cluster, which is given by the user. This algorithm works on the principle of minimizing the dissimilarities between each & every object in a cluster and its representative object. Initially, we have to choose the representative object arbitrarily and this object is known as Medoid. Find a single partition of the data into K clusters such that each cluster has a most representative point, i.e., a point that is the most “centrally” located point in the cluster with respect to some measure, e.g., distance. These representative points are called Medoids.

F. Bat Algorithm

Bat algorithm is swarm intelligence based algorithm which is worked on the echolocation of bats. This algorithm was developed by Xin-She Yang in 2010. It is a new metaheuristic algorithm for solving the many optimization problems. Bats are based on the echolocation behaviour of bats. They can find their prey o food and also they can know the different type of insects even in a complete darkness. Since we know that microbats are the insectivore who have the quality of fascinating. These bats use a type of sonar namely as echolocation. They emit a loud sound pulse and detect a echo that is comes back from their surrounding objects. Their pulse varying in properties and will be depend on the species. Their loudness is also varying. When they are searching for their prey, their loudness is loudest if they are far away from the prey and they will become slow when they are nearer to the prey. Now for emission and detection of echo which are generated by them, they use time delay. And this time delay is between their two ears and the loudness variation of echoes. The following formulae are used for their position xi and velocities vi when they are updated: fi = fmin + (fmax − fmin)β, ...(2.1.1) vti = vt-1i + (xti – x*)fi, ...(2.1.2) xti = xt-1i + vti , ...(2.1.3) We know that loudness is decreases when bat found its prey or food but the rate of pulse emission increases. For simplicity we use loudness A0 = 1 and minimum Amin = 0 that means a bat found their prey and they stop making sound. where β ∈ [0, 1] is a random vector drawn from a uniform distribution. Here x∗ is the current global best location (solution) which is located after comparing all the solutions among all the n bats. Generally speaking, depending on the domain size of the problem of interest, the frequency f is assigned to fmin = 0 and fmax = 100 in practical implementat ion. Initially, each bat is randomly given a frequency which is drawn uniformly from [fmin, fmax]. For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk. Xnew = xold + εAt ...(2.1.4) The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization as fi in essence controls the pace and range of the movement of the swarming particles. To some degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate. Furthermore, the loudness Ai and the rate ri of pulse emission update accordingly as the iterations proceed

G. Proposed K Mediods Bat Algorithm

In this section we will speed up local search by relaxing the number of clusters in the intermediate steps and achieve exactly k clusters in the final step. We must use at least k medians in the intermediate steps, since the best solution with k - 1 medians can be much more expensive than the best k median solution, and we are interested in guarantees. At the same time we cannot use too many since we want to save space. Since we have flexibility in k, we can develop a new cluster head detection using bat algorithm.
The proposed algorithm is given in as:

```
•  Initializing the objective function f(x)
•  Initialize the population of bat xi where i=1,2,3....n and
   velocity vi
•  Set the maximum iteration max_iter
•  Set time t=1
•  Define pulse frequency fi for bats at each position
•  Define the pulse rate that is emitted by the bat ri and the
   loudness Ai
   o  While(t < max_iter)
   o  Generate the new solutions by adjusting frequencies
      and update velocity and location using formulae
      (equation (1) to (3))
   o  Initialize k as initial representative object
   o  Assign each xi to its nearest representative object
      with the most similar based on the medoids of the
      objects in a cluster
   o  Select a non representative object randomly Orandom
   o  Calculate the cost S
If S < 0
Swap the Oi with Orandom
Else
Go to step b
If ( rand > ri)
•  Select a solution from the best solutions
•  Generate a local solution
•  End if
   o  Generate a new solution by the flying of bats
      ramdomly
   o  If (rand <Ai)
•  Accept the new solution
•  Increment the pulse rate and decrement the loudness
   o  End if
   o  Rank the bats and find the bet population in bat x*
   o  Increase the t
   o  End while
•  Result will be given
```

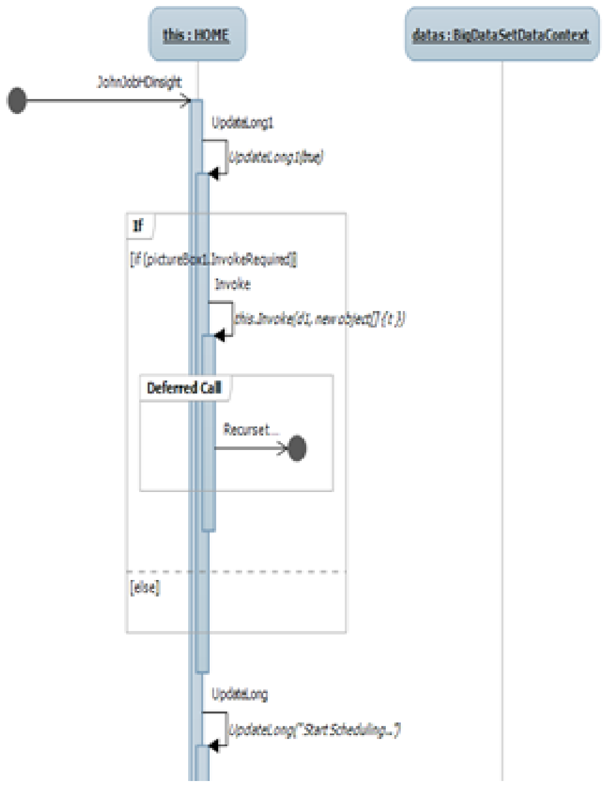
this : HOME
datas : BigDataSetDataContext
JohnJobHDinsight
UpdateLong1
UpdateLong1(true)
If
[if (pictureBox1.InvokeRequired)]
Invoke
this.Invoke(d1, new object[] { t })
Deferred Call
Recurset...
[else]
UpdateLong
UpdateLong("Start Scheduling...")

Fig. 8. Clustering flow diagram.

```
Algorithm Initial Cluster (data set N , clusters z )
1. Reorder data points randomly
2. Create a cluster center at the first point 3. For every point after the first,
    • Let d= distance from the current point to the nearest existing cluster center
    • With probability d/z create a new cluster center at the current point; otherwise add the current point to the best current cluster
```

This algorithm runs in time proportional to n times the number of facilities it opens and obtains an expected k-approximation to optimum [11].

IV. Result Evaluation

This section presents evaluation results of IAPKM and IAPNA on Facebook Social dataset, which is streamed from big data, is used. The goal of this project is to use incremental variant of KM-BAT clustering, which can achieve comparable clustering performance with traditional KM-BAT by adjusting the current clustering results according to new arriving objects. Therefore, traditional KM-BAT clustering is implemented to provide benchmark-clustering performance. The benchmark results are shown in below table I,

TABLE I. Comparison results of KM-BAT vs IAPKM vs IAPNA.

Time	Object	KM-BAT	IAPKM	IAPNA
5	1500	10	12	13
10	2500	14	15	17
15	3500	18	22	27

V. Conclusion

While the concept of incremental queries has been known for some time, the clustering implications have not been explored with users. In particular, it has been an open question whether data analysts would be comfortable interacting with confidence intervals. We hope that showing the utility of these approximations will encourage further research on both the front- and back-ends of these systems. This project had shown that it is both progressive fetching and clustering to support incremental query interactions for data analysts. With such mechanisms in place, analysts can take advantage of the immediate feedback afforded by incremental queries by rapidly refining their queries, and more importantly, exploring new avenues which they would not have done before .Our approach has validated the concept of incremental queries. We have shown that it is possible to use interaction strategies that analysts have desired, but not been able to pursue given the time required to complete clustering of large scale BIG data databases. The experience of exposing users to incremental queries and approximate clustering's motivates several lines of future work. First, it has highlighted the importance of exploring representations of confidence. While error bars are conventional, they are not necessarily easily comprehensible. In addition, they can only highlight one probability value at a time. The downsides of error bars, such as the difficulties they raise with scaling, argue that there could be an opportunity to find new ways to represent confidence intervals.

References

[1] J. H. Par, "Differences among university students and faculties in social networking site perception and use: Implications for academic library services," *The Electronic Library*, vol. 28, issue 3, pp. 417-31, 2010.

[2] A. Y. Mikami, D. E. Szwedo, J. P. Allen, M. A. Evans, and A. L. Hure, "Adolescent peer relationships and behaviour problems predict young adults' communication on social networking websites," *Developmental Psychology*, vol. 46, issue 1, pp. 46-56, 2010.

[3] K. Subrahmanyam, S. M. Reich, N. Waechter, and G. Espinoza, , "Online and offline social networks: Use of social networking sites by emerging adults," *Journal of Applied Developmental Psychology*, vol. 29, issue 6, pp. 420-33, 2008.

[4] T. A. Pempek, Y. A. Yermolayeva, and S. L. Calvert, "College students social networking experiences on Facebook," *Journal of Applied Developmental Psychology*, vol. 30, issue 3, pp. 227-38, 2009.

[5] M. A. Shaheen, "Use of social networks and information seeking behaviour of students during political crises in Pakistan: A case study," *The International Information & Library Review*, vol. 40, issue 3, pp. 142-47, 2008.

[6] A. Keenan and A. Shiri, "Sociability and social interaction on social networking websites," *Library Review*, vol. 58, no. 6, pp. 438-50, 2009.

[7] U. Pfeil, R. Arjan, and P. Zaphiris, "Age differences in online social networking: A study of user profiles and the social capital divide among teenagers and older users in MySpace," *Computers in Human Behaviour*, vol. 25, no. 3, pp. 643-54, 2009.

[8] B. Babcock, M. Datar, and R. Motwani, "Sampling from a moving window over streaming data," *Proceedings SODA*, 2002.

[9] Y. Bartal, M. Charikar, and D. Raz, "Approximating min-sum □ -clustering in metric spaces," *Proceedings STOC*, 2001.

[10] A. Borodin, R. Ostrovsky, and Y. Rabani, "Subquadratic approximation algorithms for clustering problems in high dimensional spaces," *Proceedings STOC*, 1999.

[11] P. S. Bradley, U. M. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," *in Proceedings KDD*, pp. 9–15, 1998.

[12] M. Charikar, S. Chaudhuri, R. Motwani, and V. R. Narasayya, "Towards estimation error guarantees for distinct values," *in Proceedings PODS*, pp. 268–279, 2000.

[13] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in Proceedings *STOC*, pp. 626–635, 1997.

[14] M. Charikar and S. Guha, "Improved combinatorial algorithms for the facility location and k-median problems," *in Proceedings FOCS*, pp. 378–388, 1999.

[15] M. Charikar, S. Guha, E´. Tardos, and D. B. Shmoys, "A constant factor approximation algorithm for the k-median problem," *Proceedings STOC*, 1999.

[16] F. A. Chudak, "Improved approximation algorithms for uncapacitated facility location," *Proceedings IPCO, LNCS*, vol. 1412, pp. 180–194, 1998.

[17] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *Proceedings SODA*, 2002.

[18] P. Drineas, R. Kannan, A. Frieze, and V. Vinay, "Clustering in large graphs and matrices," *Proceedings SODA*, 1999.

[19] M. Ester, H. Kriegel, J. Sander, and X. Xu, A density-based algorithm for discovering clusters in large spatial databases, *In Proceedings KDD*, pp. 226–231, 1996.

[20] F. Farnstrom, J. Lewis, and C. Elkan, "True scalability for clustering algorithms," *in SIGKDD Explorations*, 2000.