IT UNIVERSITY OF COPENHAGEN

Clustering Player Behaviors in Data Streams with K-Means in Map-Reduce

by

Sigurdur Karl Magnusson

A thesis submitted in partial fulfillment for the degree of Master of Science in IT

in the
Computer Science
Software Development and Technology

Supervisors
Julian Togelius, Rasmus Pagh

April 2013

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius – and a lot of courage – to move in the opposite direction."

Albert Einstein

Abstract

The abstract is a short summary of the thesis. It announces in a brief and concise way the scientific goals, methods, and most important results. The chapter "conclusions" is not equivalent to the abstract! Nevertheless, the abstract may contain concluding remarks. The abstract should not be discursive. Hence, it cannot summarize all aspects of the thesis in very detail. Nothing should appear in an abstract that is not also covered in the body of the thesis itself. Hence, the abstract should be the last part of the thesis to be compiled by the author.

A good abstract has the following properties: *Comprehensive:* All major parts of the main text must also appear in the abstract. *Precise:* Results, interpretations, and opinions must not differ from the ones in the main text. Avoid even subtle shifts in emphasis. *Objective:* It may contain evaluative components, but it must not seem judgemental, even if the thesis topic raises controversial issues. *Concise:* It should only contain the most important results. It should not exceed 300–500 words or about one page. *Intelligible:* It should only contain widely-used terms. It should not contain equations and citations. Try to avoid symbols and acronyms (or at least explain them). *Informative:* The reader should be able to quickly evaluate, whether or not the thesis is relevant for his/her work.

An Example: The objective was to determine whether . . . (*question/goal*). For this purpose, . . . was . . . (*methodology*). It was found that . . . (*results*). The results demonstrate that . . . (*answer*).

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor…

Contents

Abstract **ii**

Acknowledgements **iii**

List of Figures **vi**

List of Tables **vii**

Abbreviations **viii**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Problem Statement 3
- 1.3 Contributions 3
- 1.4 Project Outline 3

2 Background Theory **4**
- 2.1 Player Behaviour 4
  - 2.1.1 Game Metric 4
  - 2.1.2 Features 4
- 2.2 Clustering 4
  - 2.2.1 K-Means 4
  - 2.2.2 Streaming and Data Streams 4
- 2.3 Map-Reduce and Hadoop 4

3 Related Work **5**
- 3.1 Clustering Player Behaviors 5
- 3.2 Clustering Large Data 7
  - 3.2.1 Related work vs the study 10

4 Methodology **11**
- 4.1 Game Events Data 11
  - 4.1.1 Building Features 11
- 4.2 K-Means in Map-Reduce 11
  - 4.2.1 Euclidean distance 11
  - 4.2.2 Normalized Histograms 11
- 4.3 Experimental Set-Up 11

5 Results and Discussion **12**
5.1 Results 12
5.2 Discussion 12

6 Conclusions **13**

A Appendix Title Here **14**

Bibliography **15**

List of Figures

List of Tables

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere

Chapter 1

Introduction

1.1 Motivation

We live in a world where data is being generated and stored at an amazing rate everywhere around us. Data can describe characteristics of e.g. Internet activities, social interactions, user actions and behaviors in games, scientific experiments and measurements from different devices and sensor equipments. Amount of data that is being registered and logged around us is growing in volume and complexity and storing the data in large databases or in more commonly on-line storage model called cloud storage just gets cheaper and cheaper. We live in a world of data and we are just at the beginning of its existence.

"Information is the oil of the 21st century, and analytics is the combustion engine."

Peter Sondergaard, senior VP at Gartner

In digital games, data about in-game user interactions have been logged and behaviors analyzed since the first game came out. Analyzing the user experience and behaviors of players have mostly been done in laboratories in the past, both during game development and after game launch to see if the game was played as designed. Game designs have becoming increasingly complex in the recent years offering much more freedom to the players by increasing the number of actions available, items to interact with and on-line massive multi-player persistent worlds that continue to exist after a player exits a game [1, 2]. This complexity generates much more user-centric data than before and is increasingly challenging when evaluating game designs [3, 4]. The user interactions being registered is called *user telemetry* and is translated to *game metrics* as referred in game development, providing detailed and objective numbers, e.g. total playtime, monsters killed, puzzles solved.

Collecting user's telemetry can give very detailed quantitative information on player behavior and using data mining techniques can supplement traditional qualitative approaches with large-scale behavioral analysis [5], for example show where users are getting stuck and finding actionable behavioral profiles [1, 2, 6]. In the recent years user behavior analysis have in part been driven by the emergence of massive multi-player on-line games (MMOG) and Free-to-Play (F2P) games which can have millions of users and objects that can form highly complex interactions. These game models, especially of persistent nature, are constantly monitoring users actions and their behaviors by driving their revenue with subscriptions or offer players to buy virtual items via micro transactions [1, 2, 4, 7].

One way of doing a behavioral analysis is use an unsupervised machine learning technique called clustering. Cluster analysis is a popular exploratory data mining technique that groups set of data objects together in a cluster that are more similar to each other than data objects in other groups [8]. Human beings categorizes or classifies a new object or a phenomenon based on similarity or dissimilarity of the object's descriptive features and is one of most primitive activies of humans [9]. Clustering explores the unknown patterns of the data and provide compressed data representation for large-scale data. In computer games cluster analysis or behavioral categorization can find behavioral profiles that are actionable and give high valuable insights into the game development as well as increasing the monetization [10, 11].

Most clustering algorithms are designed for modern sizes of datasets where the whole data can fit into memory or allows few passes into a database (where each data object is read more than once). It can be very expensive analyzing large-scale datasets and to get answers efficiently then one needs to reduce the set of data to be analyzed, e.g. sample fewer players and have fewer features (dimensions) to be compared. Computations for large-scale data takes time and needs to be distributed to be able to complete in reasonable amount of time. Google's MapReduce programming model was introduced in 2004 [12] and allows automatic parallelization and distribution of computations on large clusters of commodity computers. Allowing programmers and researchers to easily implement highly scalable algorithms to process large amount of data using the MapReduce model without worrying about handling failures and distributing the data with a large amount of complex code.

1.2 Problem Statement

How are the player behaviors in game XXX and how can these findings aid game development?

Considering the massive size of user telemetry data being logged and processed and the complexity of game designs there is a knowledge gap when it comes to analyzing such large-scale data efficiently.

GameAnalytics.com is a cloud hosted service for collecting, analyzing and reporting game metrics. Working with large quantities of game metric data that needs to be analysed and processed efficiently.

A valuable application for GA is to design and implement a scalable streaming version of a clustering algorithm. That can read a large data in mini-batches into memory and cluster the data incrementally.

The goal would be to incrementally find clusters efficiently in a streaming data, showing predominant characteristics of human behavior, e.g. the hardcore players, casual players, people who didn't understand the game, etc.

1.3 Contributions

- Clustering Player Behaviours using Map-Reduce framework
- Software and knowledge to GameAnalytics for further development

1.4 Project Outline

Chapter 2

Background Theory

2.1 Player Behaviour

2.1.1 Game Metric

2.1.2 Features

2.2 Clustering

2.2.1 K-Means

Many clustering methods exists but one of the most popular ones is called *k-means* [13], where k-means groups the data into k partitions or clusters and gives insights into the general distribution in those clusters. The objective function in k-means is to minimize the squared error distance to its clusters centers. *TODO* more about k-means.

K-means algorithm have been shown to be very useful in behavioral analysis to give good insights in the general behaviors found in a game [CITE].

2.2.2 Streaming and Data Streams

2.3 Map-Reduce and Hadoop

Chapter 3

Related Work

Clustering player behaviors and processing large data sets have been researched actively in the recent years. In our work we find the general player behaviors in a the case study using k-means clustering algorithm in the MapReduce framework for high scalability and parallel processing of large-scale data. The behavior of the data can change from day to day like when dealing with an endless stream of data is also discussed. In subsequent sections some of the recent and related work are given a short introduction.

3.1 Clustering Player Behaviors

Many researches have been done on clustering and predicting player behaviors over the last years to get a better understanding which kind of user behaviors are to be found when playing a game that can be actionable for game developers [2, 6, 14–16]. User behavior analysis has becoming increasingly popular in the recent years because of rise of the free-to-play (F2P) genre games in *Facebook* and *Google Play* where populations can be in millions creating complex game interactions [1, 2]. Playing these games are free and many are of persistent nature where the world in the game continues when a player exits. To be profitable these games drive their revenue via micro transactions, e.g. players buying upgrades or virtual items in game for real money [1, 2, 4, 7]. Major game publishers have also been collecting and analyzing large scale of behavior telemetry data but details of their methods are kept confidential [5, 17]. Most available research work is case-based where a specific algorithm is applied to a specific game and commercial game data sets has only become accessible recently for academic researchers [5].

Predicting player behavior in a major commercial game Tomb Raider: Underworld (TRU) was presented in a study of Drachen et al. [10]. Authors classified 1365 players in a moderate data set into four user behavioral groups using six statistical gameplay features based on core game design as inputs of an emergent self-organizing map to identify dissimilar behavior clusters. Behavior profiles covering 90 percent of the users in the dataset were labeled in game terminology usable for game designers. Mahlmann et al. [11] did a follow up on the research using eight gameplay features and classified behavior of 10,000 players. The authors presented also how to predict behavior based on early play analysis, a popular topic which can be used to prevent churn (attrition) [7].

Analyzing social groups in the highly popular Massively Multiplayer Online Role-Playing Game (MMORPG) World of Warcraft was done by Thurau and Bauckhage [18]. They analyzed how groups (guilds) evolve over time from both American and European based guilds. Their paper is the first study analyzing such amount of data in a MMORPG, analyzing large-scale data gathered on-line from 18 million players belonging in 1.4 million groups over a period of 4 years. Convex-Hull Non Negative Matrix Factorization (CH-NMF) [16] technique was applied to the data to find the extremes rather than averages and the results show no significant cultural difference in formation processes of guilds from either the US or the EU. Interpretability of CH-NMF was more distinguishable and representing archetypal guilds than the more conventional clustering method k-means that represent the cluster centroids with similar characteristic.

Drachen et al. [6] did a clustering analysis for two major commercial games applied to large-scale of high-dimensionality player behavior telemetry. K-means and Simplex Volume Maximization (SIVM) clustering were applied to the MMORPG *Tera* and the multi-player first-person shooter strategy game *Battlefield: Bad Company 2*. SIVM clustering is an adaption of Archetype Analysis (AA) for large-scale data sets to find extreme player behaviors profiles [16, 19]. The authors show the contribution differences from the two algorithms where k-means gives insights into the general distribution of behaviors vs. SIVM showing players with extreme behaviors. The selection of the most important features from the data set were followed by a method suggested by Drachen et al. [10], behavioral profiles were extracted and interpreted in terms of design language [10, 17].

In a recent study by Drachen et al. [20] the authors compare four different popular methods with purpose of clustering player behaviors and develop profiles from large-scale game metric data set from the highly popular commercial MMORPG World of Warcraft. The data set was collected from mining the Warcraft Realms site, recordings of on-line time and what level each player reached for each day in the years 2005-2010 for approx. 70 thousands of players. The authors selected playtime and leveling speed as their behavioral variables to show a measure of the overall player engagement in the game, where playtime is one of the most important measure for calculating the churn rate [4, 7]. Interpretable behaviors profiles where only generated by the k-means and the SIVM algorithm. The SIVM Archetype Analysis algorithm produces however significantly different behaviors that result in easier interpretation of behavior profiles compared to the k-means algorithm where the centroids are overall similar.

3.2 Clustering Large Data

K-means is one of the most studied clustering algorithm out there and is still actively researched. It's a simple algorithm that partition the data into k partitions by minimizing its objective function sum of squared error. From its appearance in a standard algorithm version by Stuart Lloyd in 1982 [13], it has been one of the most popular clustering algorithm to research because of its simplicity. There are many different research areas regarding k-means e.g. manually set the number of k partitions to cluster, initializing the centers of the partitions, dealing with outliers in data and scalability when dataset increases. In recent years k-means has also been very popular algorithm to study in the Map-Reduce framework where the algorithm can easily be applied to cluster large amount of data sets in parallel [12].

Guha et al. [21] designed an algorithm called STREAM that is based on the divide-and-conquer strategy and achieves a constant-factor approximation solving the k-median problem (a k-means variant). The algorithm divides the dataset into m pieces of similar sizes. Each of the pieces are independently clustered sequentially and all the centers from all the pieces are then clustered further. They show a new k-median algorithm called LSEARCH that is used by the stream algorithm and is based on local search algorithm solving the facility location problem [22] to solve the k-median problem. Results show that LSEARCH produced better quality clusters than k-means and the hierarchical algorithm BIRCH [23] but took longer to run. In 2009 Ailon et al. [24] extended the the work of Guha et al. introducing an one pass streaming algorithm for k-means with approximation guarantees. Achieving that they introduced a new algorithm called k-means# that builds on the non-streaming algorithm k-means++ by Arthur and Vassilvitskii [25] that is a combined algorithm of seeding the initial centers and running k-means. k-means# provides a bi-criterion (α, β) approximation algorithm by choosing $\alpha * k$ centers with approximation factor β. In the divide-and-conquer strategy they run the k-means# independently on each piece of the data to achieve $O(k * log(k))$ random centers non-uniformly and use the k-means++ algorithm to find k centers from the intermediate centers from all the pieces of the data set.

Another approach is using a coreset by selecting a weighted subset from the original data set such that by running any k-means algorithm on the subset will give near similar results to running k-means on the original data set. Ackermann et al. [26] introduce a new algorithm called StreamKM++ that uses k-means++ algorithm from Arthur et al. [25] to solve k-means on the subset and the also design a new data structure called coreset tree to speed up the time for the sampling in the center initialization. StreamKM++ is a streaming version of k-means++ to cluster large data sets. Their approach was shown to be on par with LSEARCH algorithm in cluster quality but outperformed BIRCH by factor of 2. A recent work by Shindler et al. [27] proposed an algorithm called *Fast streaming k-means* based on the online facility location algorithm [28] and extends the work of Braverman et al. [29] proving a faster running time and a better approximation factor. Their algorithm outperformed both work of Ailon et al. [24] and Ackermann et al. [26] mentioned above.

Clustering data streams of an unknown length, evolving over time [30, 31] are challenging where it is not possible to access historic data points because of the amount of data arriving continuously. Aggarwal et al. [32] proposed a well-known stream clustering framework called CluStream for clustering large evolving data streams and is guided by application-centered requirements. CluStream has an online component that maintains snapshots of statistical information about micro-clusters (a.k.a. *cluster feature vector* [23]) in a pyramidal time window and an offline component that uses the compact intermediate summary statistics from the micro-clusters to find higher level k clusters using k-means, in a time horizon defined by an analyst. The authors proposed also a new high-dimensional, data stream clustering algorithm called HPStream that is highly scalable [33]. HPStream uses projected clustering [34], which can determine clusters for a subset of dimensions, to data streams and a new data structure called *fading cluster structure* that allows historical and current data to integrate nicely with a user-specified fading factor. Zhou et al. [35] presented a clustering algorithm called SWClustering that clusters evolving data streams over sliding windows to be able to analyze also the evolution of the individual clusters by eliminating influence by historic data points while the new data points arrive. The Authors show that the CluStream algorithm is more sensitive to influences of outdated data and is less efficient.

Processing large of amount of data efficiently using parallel processing is an active research and gain much of popularity when the Google's MapReduce programming model was introduced by Dean and Ghemawat [12] that allows researchers easily to create highly scalable and fault tolerance algorithms by just implementing a Map and a Reduce function. Zhao et al. [36] implemented a parallel version of k-means (PKMeans) in MapReduce framework and show that their algorithm can effectively run on large data sets. They designed the Map function to calculate the closest cluster centroid for each point at a time and output a $< key, value >$ pair where key is the closest cluster centroid id and value is the data point. After each map task they apply a combiner function called Combine that is executed on the same machine as the Map function. The Combine function partly sum the values of the data points assigned to the same cluster and outputs a $< key, value >$ pair where key is the id of the cluster centroid and the value is comprised of the sum values and the number of data points in that cluster. Using a combiner reduces the amount of intermediate information that is processed and sent over the network to the Reduce task [12, 36]. Then the Reduce function sums up all the intermediate sub sum values from all the combiners and calculates the new centers for the next MapReduce iteration.

To overcome the sensibility of k-means to outliers in data, Li et al. [37] proposed an algorithm called MBK-means using the ensemble learning method called Bagging [38], generating k new data sets from the original data set with replacement sampling. For each of the new data set the k-means is run using the MapReduce framework until convergence then finally the k sets of k centroids are merged to form the final k centroids.

Many extensions on the traditional MapReduce framework have been proposed to support algorithms running iteratively [39–44] and incrementally [44–46] efficiently. The incremental MapReduce frameworks are interesting and relates to our work since we are incrementally clustering data but the use of these frameworks is not in this study.

3.2.1 Related work vs the study

Our work is most similar to PKMeans [36] described above, a parallel k-means implementation in MapReduce using a Combine function to reduce the intermediate data sent between the mappers and the reducers. We implement a parallel k-means algorithm in MapReduce so that each Map function efficiently calculates the distance to nearest cluster centers by calculating the distance matrix between all data points and the cluster centers instead of processing each data point separately. We show the algorithm in application where it incrementally clusters player behaviors where theoretically large data sets arrive daily. In our work we show k-means is a good approach to provide insights into the general distribution of behaviors in a game, work of Drachen et al. [6, 20] relates to ours when selecting and building behavioral variables from user telemetry and extracting behavioral profiles by analyzing and interpret the centroids (basis vectors) in k-means.

Chapter 4

Methodology

4.1 Game Events Data

4.1.1 Building Features

4.2 K-Means in Map-Reduce

4.2.1 Euclidean distance

4.2.2 Normalized Histograms

4.3 Experimental Set-Up

Chapter 5

Results and Discussion

5.1 Results

5.2 Discussion

Chapter 6

Conclusions

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] Jun H. Kim, Daniel V. Gunn, Eric Schuh, Bruce Phillips, Randy J. Pagulayan, and Dennis Wixon. Tracking real-time user experience (true): a comprehensive instrumentation solution for complex systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 443–452, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357126.

[2] Anders Drachen and Alessandro Canossa. Evaluating motion: Spatial user behaviour in virtual environments. *International Journal of Arts and Technology*, 4 (3):294–314, 2011. doi: 10.1504/IJART.2011.041483.

[3] Randy J. Pagulayan, Kevin Keeker, Dennis Wixon, Ramon L. Romero, and Thomas Fuller. User-centered design in games. In Julie A. Jacko and Andrew Sears, editors, *The human-computer interaction handbook*, chapter User-centered design in games, pages 883–906. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003. ISBN 0-8058-3838-4. URL `http://dl.acm.org/citation.cfm?id=772072.772128`.

[4] M Seif El-Nasr and Canossa A Drachen A. *Game Analytics: Maximizing the Value of Player Data*. Springer, 2013. ISBN 978-1-4471-4768-8.

[5] Geogios N. Yannakakis. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, CF '12, pages 285–292, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1215-8. doi: 10.1145/2212908.2212954.

[6] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 163–170, 2012. doi: 10.1109/CIG.2012.6374152.

[7] Tim Fields and Brandon Cotton. *Social Game Design: Monetization Methods and Mechanics*. CRC Press, 12 2011. ISBN 978-0240817668.

[8] Rui Xu and II Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005. ISSN 1045-9227. doi: 10.1109/TNN.2005.845141.

[9] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[10] A. Drachen, A. Canossa, and G.N. Yannakakis. Player modeling using self-organization in tomb raider: Underworld. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 1–8, 2009. doi: 10.1109/CIG.2009.5286500.

[11] T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, and G.N. Yannakakis. Predicting player behavior in tomb raider: Underworld. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 178–185, 2010. doi: 10.1109/ITW.2010.5593355.

[12] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1251254.1251264`.

[13] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489.

[14] Tim Marsh, Shamus Smith, Kiyoung Yang, and Cyrus Shahabi. Continuous and unobtrusive capture of User-Player behaviour and experience to assess and inform game design and development. In *1st World Conference for Fun 'n Games*, Preston, England, 2006.

[15] Olana Missura and Thomas Gärtner. Player modeling for intelligent difficulty adjustment. In João Gama, VítorSantos Costa, AlípioMário Jorge, and PavelB. Brazdil, editors, *Discovery Science*, volume 5808 of *Lecture Notes in Computer Science*, pages 197–211. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04746-6. doi: 10.1007/978-3-642-04747-3˙17. URL `http://dx.doi.org/10.1007/978-3-642-04747-3_17`.

[16] C. Thurau, K. Kersting, and C. Bauckhage. Convex non-negative matrix factorization in the wild. In *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, pages 523–532, 2009. doi: 10.1109/ICDM.2009.55.

[17] G Zoeller. Game development telemetry. In *Proceedings of the Game Developers Conference*, 2010.

[18] C. Thurau and C. Bauckhage. Analyzing the evolution of social groups in world of warcraft ő. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 170–177, 2010. doi: 10.1109/ITW.2010.5593358.

[19] K. Kersting, M. Wahabzada, C. Thurau, and C. Bauckhage. Hierarchical convex nmf for clustering massive data. In Qiang Yang Masashi Sugiyama, editor, *Proceedings of the 2nd Asian Conference on Machine Learning (ACML–10)*, Tokyo, Japan, Nov 8–10 2010. URL `http://www-kd.iai.uni-bonn.de/pubattachments/477/kersting10acml.pdf`. draft.

[20] A. Drachen, C. Thurau, R. Sifa, and C. Bauckhage. A comparison of methods for player clustering via behavioral telemetry. In *Foundations of Digital Games 2013*, 2013.

[21] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O"Callaghan. Clustering data streams: Theory and practice. *Knowledge and Data Engineering, IEEE Transactions on*, 15(3):515–528, 2003. ISSN 1041-4347. doi: 10.1109/TKDE.2003.1198387.

[22] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 378–388, 1999. doi: 10.1109/SFFCS.1999.814609.

[23] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996. ISSN 0163-5808. doi: 10.1145/235968.233324.

[24] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. *Advances in Neural Information Processing Systems*, 22:10–18, 2009. URL `http://www1.cs.columbia.edu/~rjaiswal/ajmNIPS09.pdf`.

[25] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

[26] Marcel R. Ackermann, Christiane Lammersen, Marcus Märtens, Christoph Raupach, Christian Sohler, and Kamil Swierkot. Streamkm++: A clustering algorithms for data streams. In *ALENEX*, pages 173–187, 2010. URL http://www.siam.org/proceedings/alenex/2010/alx10_016_ackermannm.pdf.

[27] Michael Shindler, Alex Wong, and Adam W. Meyerson. Fast and accurate k-means for large datasets. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2375–2383. NIPS, 2011. URL http://books.nips.cc/papers/files/nips24/NIPS2011_1271.pdf.

[28] A. Meyerson. Online facility location. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, FOCS '01, pages 426–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1390-5. URL http://dl.acm.org/citation.cfm?id=874063.875567.

[29] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 26–40. SIAM, 2011. URL http://dl.acm.org/citation.cfm?id=2133036.2133039.

[30] Charu C. Aggarwal. An intuitive framework for understanding changes in evolving data streams. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 261–, 2002. doi: 10.1109/ICDE.2002.994715.

[31] Charu C. Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 575–586, New York, NY, USA, 2003. ACM. ISBN 1-58113-634-X. doi: 10.1145/872757.872826.

[32] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003. ISBN 0-12-722442-4. URL `http://dl.acm.org/citation.cfm?id=1315451.1315460`.

[33] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 852–863. VLDB Endowment, 2004. ISBN 0-12-088469-0. URL `http://dl.acm.org/citation.cfm?id=1316689.1316763`.

[34] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 61–72, New York, NY, USA, 1999. ACM. ISBN 1-58113-084-8. doi: 10.1145/304182.304188.

[35] Aoying Zhou, Feng Cao, Weining Qian, and Cheqing Jin. Tracking clusters in evolving data streams over sliding windows. *Knowl. Inf. Syst.*, 15(2):181–214, May 2008. ISSN 0219-1377. doi: 10.1007/s10115-007-0070-x.

[36] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 674–679, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-10664-4. URL `http://dx.doi.org/10.1007/978-3-642-10665-1_71`.

[37] Hai-Guang Li, Gong-Qing Wu, Xue-Gang Hu, Jing Zhang, Lian Li, and Xindong Wu. K-means clustering with bagging and mapreduce. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–8, 2011. doi: 10.1109/HICSS.2011.265.

[38] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 0885-6125. doi: 10.1023/A:1018054314350.

[39] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears. Mapreduce online. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 21–21, Berkeley,

CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855711.1855732.

[40] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: a runtime for iterative mapreduce. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 810–818, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-942-8. doi: 10.1145/1851476.1851593.

[41] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1863103.1863113.

[42] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. Haloop: efficient iterative data processing on large clusters. *Proc. VLDB Endow.*, 3(1-2): 285–296, September 2010. ISSN 2150-8097. URL http://dl.acm.org/citation.cfm?id=1920841.1920881.

[43] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. The haloop approach to large-scale iterative data analysis. *The VLDB Journal*, 21 (2):169–190, April 2012. ISSN 1066-8888. URL http://dx.doi.org/10.1007/s00778-012-0269-7.

[44] Cairong Yan, Xin Yang, Ze Yu, Min Li, and Xiaolin Li. Incmr: Incremental data processing based on mapreduce. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, pages 534–541, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4755-8. URL http://dx.doi.org/10.1109/CLOUD.2012.67.

[45] Pramod Bhatotia, Alexander Wieder, Rodrigo Rodrigues, Umut A. Acar, and Rafael Pasquin. Incoop: Mapreduce for incremental computations. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, SOCC '11, pages 7:1–7:14, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0976-9. URL http://doi.acm.org/10.1145/2038916.2038923.

[46] Pramod Bhatotia, Marcel Dischinger, Rodrigo Rodrigues, and Umut A Acar. Slider: Incremental sliding-window computations for large-scale data analysis. *MPI-SWS-2012-004*, September 2012. URL http://www.mpi-sws.org/tr/2012-004.pdf.