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

List of Figures

List of Tables

Abbreviations

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

Chapter 1

Chapter Title Here

1.1 Motivation

We live in a time of data, where data is growing at an amazing rate. Data is everywhere around us and companies are generating more and more data. Storing data is cheaper than ever and many smaller and mid-sized companies are buying storage subscriptions at other bigger companies that offer "Storage as a Service" (SaaS). A SaaS company offers e.g. data reliability and durability by storing critical data in multiple facilities and on multiple devices. Having all this data does not give a meaning by itself, it needs to be analysed and interpret to give a business value.

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

Peter Sondergaard, senior VP at Gartner

The need to analyse all this data have caused many new companies to rise up that are specialised in analysing large quantities of data, extracting knowledge and converting it to a business value. Many of those companies offer "Analytics as a Service" (AaaS) that offer an analytical software to discover trends and unknown patterns in the data.

1.2 Problem Statement

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

2.2.2 Streaming and Data Streams

2.3 Map-Reduce and Hadoop

Chapter 3

Related Work

In subsequent sections we will give a overview of some of the related and recent work. We start with looking into work related to Clustering Player Behaviors and then we dive into work related to clustering large set of data using the k-means algorithm, where we can have a finite stream of data, an endless and evolving data stream and finally some parallel clustering implementations in the Map-Reduce framework. In our work we focus on processing a large amount of data of finite length in parallel using Map-Reduce and incrementally cluster data arriving each day. The behavior of the data can change from day to day like when dealing with an endless stream of data but is not in the scope, related work is presented and solutions are discussed in the Conclusions section. [TODO].

3.1 Clustering Player Behaviors

Many researches have been done on clustering and predicting player behaviors over the years to get a better understanding which kind of groups of behaviors are playing a game by data mining user telemetry that is being logged as a the game is being played.

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 version by Stuart Lloyd in 1982 [1] , 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 and dealing with outliers in data. In recent years k-means has also been very popular algorithm to study in the Map-Reduce framework where the k-means algorithm can easily be applied to cluster large amount of data sets in parallel [2]. Starting with work relating clustering stream of data of length n that doesn't fit in memory, the incrementally evolving characteristics of endless data streams and recent work using the Map-Reduce framework.

We start with a famous work by Guha et al. [3] where they propose the STREAM algorithm 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 [4] to solve the k-median problem. Results show that LSEARCH produced better quality clusters than k-means and the hierarchical algorithm BIRCH [5] but took longer to run.

In 2009 Ailon et al. [6] extended the the work of Guha et al. mentioned above and showed 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 et al. [7] 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 $O(klogk)$ 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 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. [8] proposed a new algorithm called StreamKM++ that uses k-means++ algorithm from Arthur et al. [7] to solve k-means on the subset and also proposed 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. [9] proposed an algorithm called *Fast streaming k-means* based on the online facility location algorithm [10] and extends the work of Braverman et al. [11] proving a faster running time and a better approximation factor. Their algorithm outperformed both work of Ailon et al. [6] and Ackermann et al. [8] mentioned above.

A different approach was introduced by Sculley in 2010 [12], modifications to k-means for batch optimizations. An algorithm called Mini-batch k-means that yields excellent clustering results with low computational cost for large datasets. The algorithm initializes the k centers like the normal k-means algorithm but then for each iteration it picks b sized examples randomly from the dataset and for each point in the example the center which it is closest to is updated by taking a gradient descent step towards the point, with a learning rate of that center. The approach scales when datasets grow large with redundant examples and allowing convergence to better solutions compared to the on-line stochastic gradient descent (SGD) variant proposed by Bottou et al. [13].

Clustering a data stream where data arrives continuously and the behavior of the stream can change over time. This area of research is very popular in the recent years where generating data is rapid, has an unknown length and not possible to access historic data points seen before because of the amount of data.

Map-Reduce. A parallel version of k-means++ initialization algorithm can be found in the work of Bahmani et al. from 2012 [14]. The algorithm is called k-means|| and is implemented in the Map-Reduce framework and they show it outperforms the k-means++ algorithm in both sequential and parallel settings.

3.2.0.1 Map-Reduce

3.2.1 Data Streams

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] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1177–1178, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772862.

[13] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.3258.

[14] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, March 2012. ISSN 2150-8097. URL http://dl.acm.org/citation.cfm?id=2180912.2180915.