# Big Data Gathering and Mining Pipelines for CRM using Open-source

Kang Li*, Vinay Deolalikar†, and Neeraj Pradhan‡

Search and Data Mining
Groupon
Palo Alto, CA 94306
E-mail: *kli@groupon.com, †vdeolalikar@groupon.com, ‡neepradhan@groupon.com

## Abstract

*Customer Relationship Management (CRM) is currently the fastest growing sector of enterprise software, estimated to increase to $36.5B worldwide by 2017. CRM technologies increasingly use data mining primitives across multiple applications.*

*At the same time, the growth of big data has led to the evolution of an open source big data software stack (primarily powered by Apache software) that rivals traditional enterprise database (RDBMS) stacks. New technologies such as Kafka, Storm, HBase have significantly enriched this open source stack, alongside more established technologies such as Hadoop MapReduce and Mahout. Today, enterprises have a choice to make regarding which stack they will choose to power their big data applications.*

*However, there are no published studies in literature on enterprise big data pipelines built using open source components supporting CRM. Specific questions that enterprises have include: how is the data processed and analyzed in such pipelines? What are the building blocks of such pipelines? How long does each step of this processing take?*

*In this work, we answer these questions for a large scale (serving over a 100M customers) industrial CRM pipeline that incorporates data mining, and serves several applications. Our pipeline has, broadly, two parts. The first is a data gathering part that uses Kafka, Storm, and HBase. The second is a data mining part that uses Mahout and Hadoop MapReduce. We also provide timings for common tasks in the second part such as data preprocessing for machine learning, clustering, reservoir sampling, and frequent itemset extraction.*

## 1. Introduction

Today, perhaps more than ever before, enterprises realize that understanding their users is critical to their success. Accordingly, Customer Relationship Management (CRM) is currently the fastest growing sector of enterprise software, estimated to increase to $36.5B worldwide by 2017 [4]. CRM for an enterprise with a large customer base has significant big data challenges, typically involving both a streaming aspect (capturing customer behavior) and a batch component (analyzing and mining captured behavior). Also, CRM technologies increasingly use data mining techniques across multiple applications.

At the same time, the field of big data has been evolving rapidly. A highly significant evolution is the formation of a formidable open source stack of technologies, starting with Apache Hadoop, and incorporating recent entrants such as Apache Kafka and Storm. Typically today, both RDBMS and open source stacks co-exist in different parts of the enterprise big data ecosystem. Although RDBMS is still used mostly for structured data, unstructured data is typically stored on HDFS (including NoSQL databases like HBase which use HDFS under the hood). Hadoop is used for large scale batch computations (such as clustering), which would be difficult to replicate using traditional RDBMS. Hadoop is also highly scalable and we can add more nodes to the cluster to reduce computation time, something that is not the case with traditional RDMBS.

The preceding two threads bring us to the problem statement of this poster, which we pose as two questions.

1) What are the components of a big data processing and mining pipeline, using the open source stack, for powering CRM applications?
2) What are reasonable performance baselines for standard tasks using such open source components?

In this poster, we parse out concrete questions that we would like to answer.

1a What are the open source components of such a big data CRM pipeline? How do these components address the streaming and the batch aspects of the big data challenge for CRM?
1b Where are fundamental primitives such as clustering, frequent itemset extraction, and sampling located in the pipeline?
2 How long do typical data preprocessing steps and fundamental data mining primitives take, without excessive tuning (namely, using default settings)?

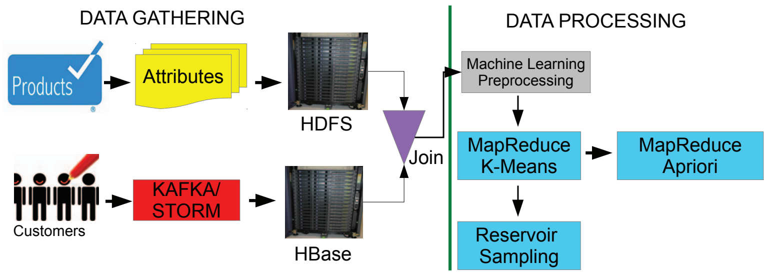

Fig. 1: The big data gathering and processing Pipeline

We present details of an industrial big data pipeline built using open source components that is currently serving CRM applications for upwards of 100M customers. We provide details on how the various technologies play their respective roles in this data pipeline. Finally, we provide performance benchmarks for standard tasks encountered in CRM (especially using data mining) on open source components using their default settings.

## 2. Related Work

In this abstract, we provide a brief overview of major streams of research that relate to our work.

**CRM.** As noted, CRM has become the fastest growing sector of enterprise software [4]. Accordingly, there are several white papers that describe CRM solutions from leading vendors such as Oracle and IBM [5, 11]. However, there is little in published research literature on end-to-end CRM solutions using open source frameworks, and including benchmarks. Our work attempts to fill that gap.

**Big Data and Open Source.** Open source has become a disruptive trend in big data, and the literature in it has become correspondingly large. [10] lists 100 papers covering various aspects of big data that are addressed by open source solutions. Most of the technical description for each open source component occurs online, and corresponding URLs are provided throughout this paper.

**Big Data Benchmarks and MapReduce Optimizations.** [6] do an in-depth performance analysis of MapReduce. [7] provide a host of big data benchmarks on various platforms, focusing on Hive and Hive-like SQL interfaces. The PUMA project[1] provides MapReduce benchmarks on various algorithmic primitives such as sort and join. In addition, there is a body of work on optimizing MapReduce for certain tasks. For example, [15] study parallel K-means and [3] optimize K-means on MapReduce.

**Query Platforms on MapReduce.** Certain platforms place a data warehousing solution on top of MapReduce that stores the data in the form of tables so that a user can query it using a restricted query language, while taking advantage of the parallelization of the MapReduce framework. A leading example is Hive [13] and its query language HiveQL, which are built on top of Hadoop. A more recent effort in this direction is Spark[2]. Despite the growth of such platforms in the recent past, a significant proportion of open source big data still happens on Hadoop MapReduce.

**Algorithmic Research on CRM.** Our work does not focus on algorithms for CRM. In this area, perhaps the most research effort in CRM has come from eCommerce, in modeling users and recommending items to them. Notable works include Amazon.com's item-item collaborative filtering [9], segmentation of customers [14], and clustering of click-streams [12]. Our work significantly differs from these works since these works represent algorithmic research, but do not discuss the big data pipelines required in order to scale these algorithms to over a hundred million users. This is the gap in literature that our work tries to address.

## 3. Two Parts of Big Data Pipeline

We will assume background on Apache Kafka[3], Apache Storm[4], Apache HBase(an open-source, distributed, versioned, non-relational database modeled after Google's Bigtable [2] ), and Hadoop MapReduce.

The study in this posted is conducted at a multi-billion dollar e-commerce company—Groupon. It comprised tens of billions of customer "activities" per month, aggregated over 100M customers of Groupon, over thousands of products. An "activity" refers to a gamut of customer interactions (product views, clicks, purchases, etc.), collected across multiple touch points (website, mobile application, curated product catalogs sent over email, etc.).

Given the massive scale of the data above, we built two big data pipelines. The first is for collecting and pre-processing such large amount of activities, and the other one is for processing and mining the resulting data. Broadly speaking, the first pipeline tackles the problem of big streaming data, while the second addresses big batch data.

### 3.1. Data Gathering and Joining Pipeline

The first part of our big data pipeline, as shown in Fig.1 (left of dashed vertical line), is our data collection pipeline. Here, two different types of data, with very different properties, are gathered together. The components of this part of our data pipeline are HDFS, Kafka, Apache Storm, and HBase.

1) First, product attributes are collected from hadoop distributed file system (HDFS). In product attributes, information about products is stored, including product prices, product locations, product taxonomies,[5] etc. Some products, such as services, have location constraints associated with them. These are also stored in HDFS.
2) Second, customer activities are collected from log files, parsed into a canonicalized format in Kafka, and are written into HBase via Storm. For example, when customers are browsing and using the product catalogs on Groupon's website, they view, click and/or make purchases. In each case, the customer ID, product ID, activity type (view, click, or purchase), and activity time are recorded.

After collecting data of deal attributes and customer activities, a mapreduce job is then used to combine each customer activity and the related deal attributes by product IDs.

### 3.2. Data Processing and Mining Pipeline

The second part of our big data pipeline is the data processing (analysis and mining) pipeline shown in Fig.1 (right of dashed vertical line). This comprises several Hadoop MapReduce tasks, outlined below. These also use the Mahout libraries.

- First, an aggregation job is used to integrate customer activities by customer ID, for a unified view of customer activities over time. In this step, to reduce sparsity caused by the large amount of products, we aggregate counts of activities on product attributes independently. Besides, time decays are added to the activity count to account for the impact of time. Moreover, these counts are normalized by product positions on the website to remove the biases caused by advertisement or product positions.

1. https://engineering.purdue.edu/ puma/datasets.htm
2. https://spark.apache.org/. Although Spark does provide a query language, it can be used to replace both streaming data (like Storm) as well as MapReduce, and is strictly faster than MapReduce across multiple benchmarks. It is fast gaining in adoption, and there is speculation that it may replace Hadoop MapReduce as the de facto batch computation engine in the future.
3. http://kafka.apache.org/
4. https://storm.apache.org/

5. Product taxonomy is a human defined and tree structured category dictionary.

| Task | Library | Mappers | Reducers | Time Taken |
|---|---|---|---|---|
| Reformat data to vectors and normalize | Custom using Hadoop MapReduce | 1146 | 0 | 5mins, 31sec |
| K-Means with $K = 100$ for 20M 150-dimensional vectors | Mahout MapReduce K-Means | 1146 | 1 | Initialization: 1mins, 45sec<br>Iteration 1: 1mins, 29sec<br>Iteration 2: 1mins, 27sec<br>Iteration 3: 1mins, 27sec<br>Iteration 4: 1mins, 27sec<br>Iteration 5: 1mins, 26sec<br>Iteration 6: 1mins, 29sec<br>Final Clustering: 1mins, 27sec |
| Extract frequent itemsets from each cluster | MapReduce Apriori | 1146 | 100 | 24mins, 47sec |
| Estimate per-strata sample size for stratified sampling | Custom using Hadoop MapReduce | 160 | 1 | 3mins, 31sec |
| Reservoir sampling from each strata | Custom using Hadoop MapReduce | 160 | 20 | 2mins, 34sec |

TABLE 1: Big data tasks performed in batch mode using Hadoop MapReduce and Mahout with default settings on a one-time dump of activities of roughly 20M users taken over a period of 6 months in 2014. Number of mappers and reducers are shown. Both numbers are chosen by default based on DFS block sizes. Time required for each task is shown.

- Second, a MapReduce K-Means [15] is applied to the aggregated data to cluster customers into $K$ groups. In the clustering, aggregated activities on deal attributes are viewed as features of customers. The cosine distance is used as the distance metric of the MapReduce K-Means.
- Third, we run a MapReduce job on each group to extract frequent itemsets. In this task, we implement MapReduce Apriori[1, 8]. In each group, the learned frequent itemset forms a "signature" that is used by multiple CRM applications.
- Finally, across all the groups, we perform a stratified reservoir sampling that yields 1M customer IDs corresponding to highly active customers that are also typical for their group. This set is used for further data science using small-scale analysis tools such as R and Python. This smaller set of users allows us to rapidly protoype research ideas.

To summarize Section 3, raw customer events are consumed from Kafka and processed into a canonicalized format. The unified stream is then written into HBase via storm. In this framework, Kafka acts as a platform to handle large volume of real time logs (a.k.a. raw customer events); and Storm acts as a reliable data transmission pipeline. The Storm/Kafka pipeline is used for real-time data stream processing for analytics. For instance, within 15 mins of a customer activity occurring, it is available for processing to our data mining pipeline because Storm continuously processes the data stream from Kafka and puts it into HBase. In contrast, Hadoop MapReduce is purely an offline engine that we use to process (analyse and mine) large quantities of static data.

## 4. Performance Benchmarks

The performance benchmarks, using default settings, are provided in Table 1 for activities from 20M customers. Each customer is represented in 150-dimensional space. The Hadoop computing cluster has 18 nodes, and it can execute up to 126 mapper tasks and 54 reducer tasks at any given time. Its heap memory size is 19.39 GB. At the time of obtaining the benchmarks of Table 1, other tasks (besides those in Table 1) running on the cluster amounted to less than 15% of its capacity, so that we may assume that almost the entire cluster was available for the tasks reported in this paper.

## 5. Conclusion

In this paper, we have attempted to fill a gap in literature by providing details of a big data pipeline using open source components that serves CRM at a large corporation with a user base exceeding 100M. We provide benchmarks for processing activities from 20M customers, using default settings in Hadoop MapReduce, for various common tasks in big data analysis.

## References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. *Proc. VLDB'94*.

[2] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *Proc. OSDI '06*.

[3] Xiaoli Cui, Pingfei Zhu, Xin Yang, Keqiu Li, and Changqing Ji. Optimized big data k-means clustering usingmapreduce. *The Journal of Supercomputing*, 70(3):1249–1259, 2014.

[4] Gartner. Market share analysis: Customer relationship management software, worldwide. *Gartner White Paper*, Feb 2014.

[5] IBM. From social media to social crm: What customers want. *IBM White Paper*, Feb 2011.

[6] Dawei Jiang, Beng Chin Ooi, Lei Shi, and Sai Wu. The performance of mapreduce: An in-depth study. *Proc. VLDB Endow.*, 3(1-2):472–483, September 2010.

[7] Berkeley AMP Lab. Big data benchmarks. *available online at https://amplab.cs.berkeley.edu/benchmark/*, last retrieved July 2015.

[8] Ning Li, Li Zeng, Qing He, and Zhongzhi Shi. Parallel implementation of apriori algorithm based on mapreduce. *Proc. SNPD'12*.

[9] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 2003.

[10] LinkedIn. 100 open source big data architecture papers for data professionals. *available online at http://tinyurl.com/pkkcuc7*, last retrieved July 2015.

[11] Oracle. Knowledge-infused customer relationship management: A game-changing investment for customer support. *Oracle White Paper*, Nov 2011.

[12] Qiang Su and Lu Chen. A method for discovering clusters of e-commerce interest patterns using click-stream data. *Electronic Commerce Research and Applications*, 2015.

[13] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive: A warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2):1626–1629, August 2009.

[14] Roung-Shiunn Wu and Po-Hsuan Chou. Customer segmentation of multiple category data in e-commerce using a soft-clustering approach. *Electronic Commerce Research and Applications*, 2011.

[15] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. *Proc. CloudCom'09*.