IT@Intel White Paper
Intel IT
Big Data
February 2014

# Using Apache Hadoop* for Context-Aware Recommender Systems

The CARS manages the massive amounts of data associated with recommendation engines and adds the intelligence of immediate contextual parameters.

## Executive Overview

**Intel IT has developed a context-aware recommender system (CARS) to address big data predictive analytics challenges involving expanding data warehouses, constantly changing contextual parameters, and fast response times.**

The CARS manages the massive amounts of data associated with recommendation engines—information filtering systems that predict the rating of products and services—and adds the intelligence of immediate contextual parameters, such as time of day, location, and weather. As a result, users receive more relevant recommendations that are based on a combination of their historical preferences and contextual parameters.

By building the CARS with Intel® Distribution for Apache Hadoop* software (Intel® Distribution) we have accomplished the following:

- **Shortened time to market.** Our production-level Intel Distribution environment provides the necessary support and meets all production service-level-agreement (SLA) and operation requirements. Its comprehensive system monitoring and logging tools as well as control authentication and authorization capabilities allow us to use existing features instead of developing our own. And its optimization of execution and tuning allows for maximum performance without significant added effort.
- **Expanded revenue-generation opportunities.** Business units (BUs) can deliver more relevant recommendations that result in higher acceptance rates, more efficient business development, and increased revenue.
- **Built a reusable recommendation engine.** The initial infrastructure took several months to build. Now that it is complete, we can repurpose the CARS for other BUs in as little as four to six weeks.

We built the initial CARS architecture for our location-based-services BU, which recommended coupons to customers using a mobile navigation application. An eight-week pilot program showed that customers using our solution exchanged 45 percent more coupons than a control group that received coupons based on location only.

Since the success of the initial pilot, the CARS has been repurposed for a sales and marketing BU that needed assistance with promoting the right product to the right customer at the right time. Because the CARS is flexible and applicable in a variety of use cases, other BUs are exploring ways to use it to take advantage of its big data predictive analytics capability.

Yonatan Dolan
Project Manager, Intel IT

Oren Razon
Technical Lead, Intel IT

## Contents

## BACKGROUND

**Recommendation engines aim to predict user preferences based on historical activity and implicit or explicit feedback. They enable applications to present the most relevant information (such as a list of products or services) to users. Many Intel business units (BUs), including location-based services and sales and marketing, rely on recommendation engines.**

Context is essential to delivering a recommendation that's relevant to a user, whether it's an advertisement, coupon, promotion, reminder, product, alert, or any other kind of content. A context-aware recommender system (CARS) can also take into account contextual parameters–time of day, location, weather, season, device characteristics, and others–in its ranking of recommendations. An entity that uses a CARS can gain a competitive advantage over its competition. Because a successful CARS requires a complex architecture that processes multiple terabytes (TB) of data and enables near-real-time usage of the models created, it's challenging to build.

Intel's location-based-services BU asked Intel IT to develop a CARS that could help deliver coupons that are most relevant to the BU's customers' immediate preferences. This BU managed a mobile application that was changing from a fee-based business-to-business model to a free business-to-consumer model. Since the BU would no longer be collecting fees, it turned to location-based advertising in the form of coupons as a new revenue source. The BU wanted a CARS solution to help increase the conversion rate, which is the ratio of coupons a customer acted on to coupons presented to the customer. An increased conversion rate would equate to increased revenue.

In an effort to strengthen the relevance of the recommendations–that is, to increase the likelihood that a customer would take action–we realized the need to augment a standard recommendation engine with contextual information. This information could help us match the recommendation to tendencies inferred from historical activity. We wanted to use more than just location as a contextual cue. Incorporating other parameters, such as weather and time of day, would require transforming TBs of structured data from a variety of sources into actionable information and knowledge.

We also wanted to develop a general solution that could be used by other Intel BUs that rely on recommendation engines. For example, Intel sales and marketing groups needed help providing customers with the right product promotions at the right time.

Once we committed to developing a CARS, we faced the following challenges:

- Developing a system quickly enough to increase revenue for the location-based-services BU
- Processing requests and responses for recommendations in less than one second, factoring in various sources of contextual data
- Integrating a complex system of data warehouses, data processing algorithms, and mobile delivery systems

## SOLUTION

**Intel IT developed a CARS that addressed a specific use case: increasing the conversion rate after coupon delivery to mobile clients. We designed the system so that it can be applied to other use cases involving big data predictive analytics. Intel® Distribution for Apache Hadoop* software (Intel® Distribution) is the cornerstone of the CARS, capable of parallel distributed processing of large data sets across clusters of computers.**

By including Intel Distribution (Figure 1) in the CARS infrastructure, we enabled a system architecture and data flow that worked for our initial use case and that will adapt to future use cases.

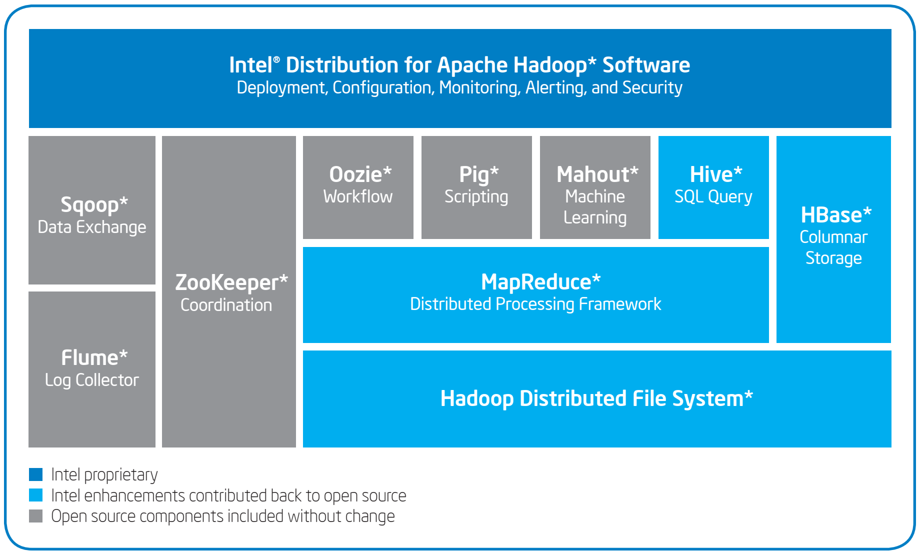
Intel® Distribution for Apache Hadoop* Software
Deployment, Configuration, Monitoring, Alerting, and Security
Sqoop* Data Exchange
Flume* Log Collector
ZooKeeper* Coordination
Oozie* Workflow
Pig* Scripting
Mahout* Machine Learning
Hive* SQL Query
HBase* Columnar Storage
MapReduce* Distributed Processing Framework
Hadoop Distributed File System*
■ Intel proprietary
■ Intel enhancements contributed back to open source
■ Open source components included without change

Figure 1. Intel® Distribution for Apache Hadoop* software (Intel® Distribution) components. Intel Distribution enables us to repurpose and reconfigure the context-aware recommender system (CARS) for additional use cases across Intel.

## The Customer Experience

The CARS was associated with a free mobile navigation application that provided information to the BU's customers (Figure 2). The BU wanted to use location-based advertising to maximize revenue by increasing the conversion rate on delivered coupons without disrupting the navigation taking place in the application. The CARS supported this goal by delivering coupons based on a customer's navigation route or single-line-search (SLS) keywords.[1]

### NAVIGATION ROUTE

When a customer traveled toward a destination, the BU gathered coupon offerings from local points of interest (POIs), such as restaurants, shops, and other businesses. The CARS then

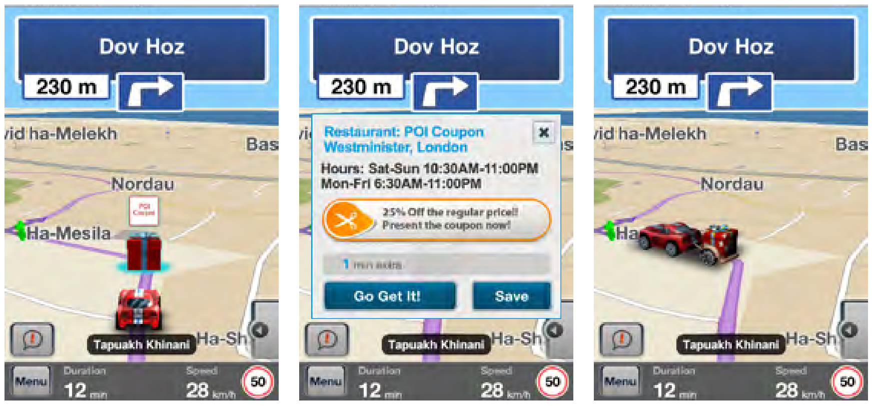
Dov Hoz
230 m
Nordau
Ha-Mesila
Tapuakh Khinani
Restaurant: POI Coupon Westminster, London
Hours: Sat-Sun 10:30AM-11:00PM Mon-Fri 6:30AM-11:00PM
25% Off the regular price!! Present the coupon now!
Go Get It!
Save
Menu
Duration 12 min
Speed 28 km/h
50
Coupon presented. Details of coupon displayed. Coupon picked up.

Figure 2. Example of a mobile navigation application delivering a coupon to a customer with the context-aware recommender system (CARS).

[1] To protect customer privacy, we masked personal information that could reveal the customer's identity. However, we did have access to navigation history and other actions performed in the application. This information was essential to creating more relevant recommendations.

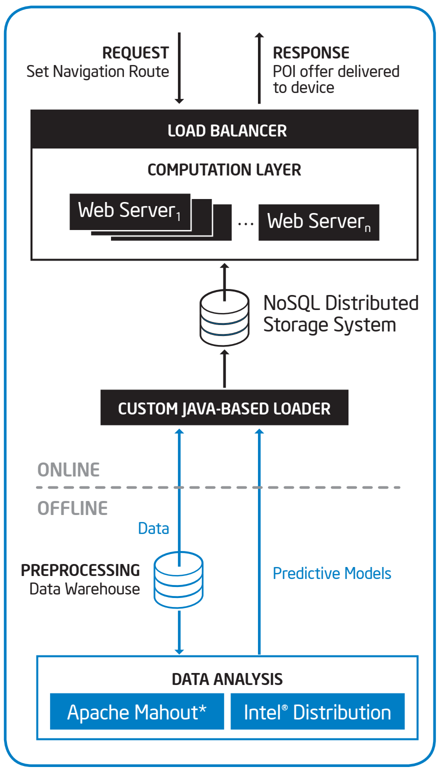

Figure 3. The context-aware recommender system (CARS) architecture. The offline process included data preprocessing and data analysis, and the online process focused on interpreting results, creating the final recommendation list.

accessed the list of potential coupons and cross-referenced it with customer-specific historical preferences in that contextual situation as well as detour variables, coupon values, time of day, and weather conditions. The CARS then processed all this information and output a list of coupons, ranked from most to least relevant, and presented them as the customer proceeded along the route.

This process—collecting the coupons, cross-referencing them against other contextual information, and presenting a ranked list—took less than one second. Customers saw coupons as they proceeded along their route and chose whether to detour to the POI, save the coupon to use later, or ignore it. To keep from overwhelming customers with coupons, the BU partitioned the route into geographic sections. The customer saw only the most relevant coupon in each section. A limit of three coupons was displayed per trip.

### SINGLE-LINE-SEARCH OPTION

When a customer entered a search phrase to find addresses or POIs, the data flow was identical to the navigation route data flow, but the CARS also factored in the SLS keywords as part of its contextual analysis and coupon ranking. The customer saw a single banner describing the most relevant coupon, along with other search results.

## System Architecture

The CARS was a specific example of a data mining process that consisted of data preprocessing, data analysis, and result interpretation. While both data preprocessing and analysis could be done offline through batch processing, the result interpretation—creating the final recommendation list—had to take place online as part of the request produced by the customer navigation route or SLS keywords. Figure 3 illustrates how we divided components into offline and online processes.

### OFFLINE PROCESS

Data preprocessing and analysis occurred in the offline process, which ran weekly to compile all the data and generate the machine-learning models to be used by the online process.

Data preprocessing involved collecting data from various sources, which were then integrated by transforming the data, if necessary. Data preprocessing took place in a data warehouse built on a shared-nothing, massively parallel processing (MPP) architecture that was capable of processing enormous amounts of structured data stored in relational databases. A combination of internal and external data sources provided historical user behavior, coupon data, geospatial information, postal codes, socio-economic characteristics, date and time, and weather information for data modeling.

Data analysis featured batch processing with advanced analytics. We designed the solution to grow linearly as the amount of data grew. The solution was built using the Intel Distribution platform, which allowed for parallel distributed processing of large data sets across clusters of computers using the MapReduce* programming paradigm. The location-based-services BU had more than 6 TB of data and had to plan for much more as the customer base expanded and more data sources were added.

Additional benefits of using Intel Distribution in the CARS infrastructure included its seamless integration with Intel security management, its close relationship with the open source community, and its optimization for Intel® technology, which allowed for quicker execution of the machine-learning algorithms.

Data analysis required executing sophisticated algorithms using Apache Mahout* on huge amounts of data from the preprocessing phase, which placed high memory and storage demands on the database. Mahout includes core algorithms of recommendation engines, classification, and clustering in ready-to-use, scalable Java-based libraries.

The models that were used after a long exploration phase in which numerous methods were tested included collaborative filtering and content-based filtering. Collaborative filtering made automatic implicit predictions about the interests of a single customer by collecting preferences or taste information from many customers. Content-based filtering was based on characteristics of the items that were going to be recommended; the results of these calculations were ready-made model tables which were later used by the online process.

### ONLINE PROCESS

Composition of the final ranked recommendation list occurred in the online process and included data retrieval (the custom Java-based loader and not only SQL [NoSQL] distributed storage system), a computational layer, and a standard API.

The NoSQL distributed storage system managed large amounts of structured data spread out across many commodity servers. It also provided a highly available service with tunable consistency and no single point of failure. This data source included the computed models from the data analysis module in the offline process as well as constantly growing amounts of user data and coupon data.

The NoSQL distributed storage system was loaded through a custom Java-based loader. We developed the loader to update the data in the system while maintaining full data-reading consistency through a version-control mechanism. The loader handled two datasets: the weekly results from data analysis and the most recently updated customer transaction logs from the offline database.

The computation layer executed the prediction flow described in Figure 3 and responded to a request in less than one second. This required high availability and performance, which was enabled with three dedicated web servers in the computation layer. A load balancer on top of the web servers balanced the workload equally. The final ranked coupons list was delivered as a JavaScript Object Notation (JSON) message.

## Data Flow

Figure 4 illustrates the flow of the data through three layers: prefiltering, modeling, and postfiltering. Customer engagement with the application initiated the creation of a list of recommendation candidates; in this case, coupons gathered by the BU. These coupons then flowed through the system to produce a ranked list of coupons that were as relevant as possible to the customer's preferences and contextual parameters.

### PREFILTERING

Before a final recommendation could be delivered, we needed all the coupons available for a given route. The BU assembled these potential coupons from multiple data sources after the customer activated the application, either through navigating a route or SLS. Once the potential coupons were assembled, the CARS applied filter rules using a knowledge-based business rules engine that was fueled by contextual information. Filter rules excluded coupons from the result set that didn't fit the appropriate parameters. When appropriate, we could adjust these rules quickly to support different business requirements.

### MODELING

The modeling layer consisted of a hybrid of two machine-learning algorithms—collaborative filtering and content-based filtering—which were computed in the offline process. By combining these two algorithms, we could combine results from the collaborative and content-based filtering models into one prediction model. This optimized the predictive performance of the CARS and produced a list of ranked candidates based on the best-matched items. (See System Architecture for more information on the modeling architecture.) Ranked candidates then proceeded to the postfiltering layer.

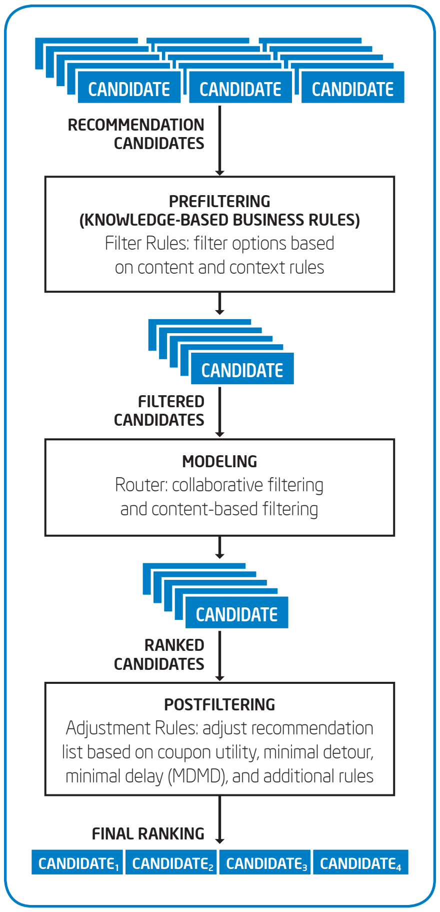
CANDIDATE
CANDIDATE
CANDIDATE
RECOMMENDATION CANDIDATES
PREFILTERING (KNOWLEDGE-BASED BUSINESS RULES)
Filter Rules: filter options based on content and context rules
CANDIDATE
FILTERED CANDIDATES
MODELING
Router: collaborative filtering and content-based filtering
CANDIDATE
RANKED CANDIDATES
POSTFILTERING
Adjustment Rules: adjust recommendation list based on coupon utility, minimal detour, minimal delay (MDMD), and additional rules
FINAL RANKING
CANDIDATE1
CANDIDATE2
CANDIDATE3
CANDIDATE4

Figure 4. Data flow. The context-aware recommender system (CARS) included three layers of data manipulation: prefiltering, modeling, and postfiltering.

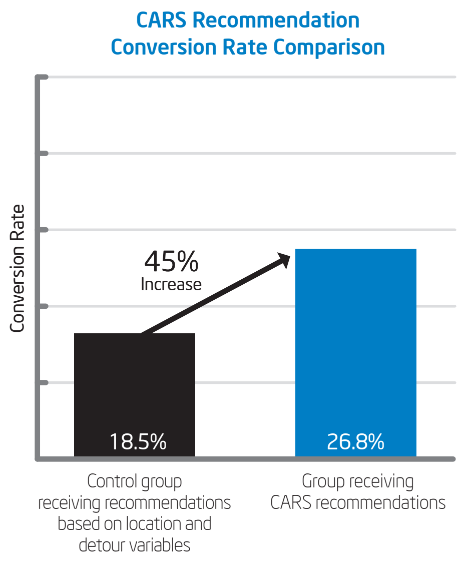

Figure 5. The group receiving the CARS recommendations showed a 45-percent higher conversion rate than the control group.

### POSTFILTERING

The CARS applied additional knowledge-based business rules to the ranked candidates, which helped adjust the scores based on current context. This produced the final ranking of the coupons and determined which coupons would be presented to the customer.

## Pilot Project

After building the complex architecture and fine-tuning the machine-learning algorithms, we conducted a pilot in the United Kingdom to evaluate the system's effectiveness. The UK's strong local customer base, combined with a mature market for the type of coupons delivered in the navigation application, gave us a dedicated population that provided valuable qualitative feedback. The population consisted of 1,100 regular customers providing quantitative feedback and a focus group of 100 Intel employees providing more immediate verbal feedback. We divided the population into two equal groups:

- One group received recommendations provided by the CARS
- A control group received recommendations based on location and detour variables only

The pilot lasted for eight weeks to allow for sufficient data collection and to let the CARS machine-learning component improve its performance based on real feedback.

### RESULTS

During the pilot, customers performed more than 17,000 navigations, and the CARS presented more than 25,000 coupons. Based on a conversion rate comparison between the two groups, the group receiving the CARS recommendations showed a 45-percent higher conversion rate than the control group (see Figure 5). Confirming the system's ability to learn, we recorded a 4-percent increase in conversion rate from the halfway point of the pilot to its conclusion. Results of this magnitude helped win approval to roll out the CARS to the full customer base of the location-based-services BU.

Additionally, we achieved our goal of providing a full ranked list of coupons within one second after the customer activated the mobile navigation application. We also delivered the CARS in less than eight months—a fast turnaround for a solution this complex. Intel Distribution and open source technology made it easier to use existing components wherever possible and adjust them to our use case, which helped limit time-consuming custom development.

## REPURPOSING THE CARS FOR ADDITIONAL USE CASES

**Now that the CARS infrastructure is in place, we can easily and quickly apply it to other use cases by identifying the appropriate data sources, reconfiguring the machine-learning models to account for those sources, and updating the knowledge-based business rules. What initially took several months to develop can now be repurposed for other use cases in as little as four to six weeks.**

We recently adapted the CARS to help Intel sales and marketing improve product purchase recommendations to channel customers, thus increasing sales and revenue. Other BUs are exploring ways to use it to take advantage of big data predictive analytics wherever software is responsible for providing users with a single recommendation from a variety of options.

## CONCLUSION

**We created a CARS that uses big data predictive analytics to increase the relevance of recommendations, providing a more satisfying user experience and helping one of Intel's BUs achieve its revenue goals more quickly. We now have the infrastructure in place to apply the CARS to other enterprise use cases within several weeks.**

Intel Distribution and its parallel distributed processing of large data sets across clusters of computers contributed significantly to the successful build of the CARS. Because Intel Distribution is optimized for Intel technology, it executed the machine-learning algorithms more quickly. Without Intel Distribution, we might have developed a solution to help the location-based-services BU, but it would have taken longer, and we would not have had the agility to reuse the infrastructure on additional use cases.

## RELATED INFORMATION

**Visit www.intel.com/IT to find content on related topics:**

- "Creating Business Value through Context-Aware Computing"
- "Integrating Apache Hadoop* into Intel's Big Data Environment"
- "Integrating Data Warehouses with Data Virtualization for BI Agility"
- "Using Big Data Predictive Analytics to Optimize Sales"

## CONTRIBUTORS

Chen Admati
Program Manager, Advanced Analytics
Intel IT

Moty Fania
Principal Engineer, Advanced Analytics
Intel IT

Abraham Israeli
Data Scientist, Advanced Analytics
Intel IT

## ACRONYMS

| | |
|---|---|
| BU | business unit |
| CARS | context-aware recommender system |
| JSON | JavaScript Object Notation |
| MDMD | minimal detour, minimal delay |
| MPP | massively parallel processing |
| NoSQL | not only SQL |
| POI | point of interest |
| SLA | service-level agreement |
| SLS | single line search |
| TB | terabyte |

**For more information on Intel IT best practices, visit www.intel.com/IT.**

Look Inside.™