Big Data Technology
NoSQL Databases Motivation:
Using Key-Value Stores to Profile Users

Eshcar Hillel
Yahoo!

Ronny Lempel
Outbrain

Applications Requiring Scalable OLTP

- OLTP – Online Transaction Processing
- Communication
  - Instant messaging/ Microblogging/ Location check-in
- E-commerce
  - Product carts
- Content recommendation, online advertising
  - User profile aggregation and serving
- …and more!

April 17, 2016
236620 Big Data Technologies
2
The NoSQL Approach to Scalable OLTP

- Traditional databases with full SQL support are very difficult to scale while preserving low latency
- NoSQL databases trade off most functionality of full-blown SQL for extreme scalability & low latency
  - Simplified semantics
  - Eliminated transactions
- Simplest manifest: super-scalable key-value store service
  - API: lookup (get), insert/update/delete (put)
  - Optimized for write-intensive workloads

Motivating Example: Building and Serving User Profiles at Outbrain

- Slides based on invited talks given at WSDM’2016 (Industry and VC Day) and BIG’2016
- Describe a true use case in the industry
- Explain what is Outbrain, why user profiles are needed, how they are built and how they are served
- Implementation issues to solve scale challenges
What Is Outbrain?

INTERNAL DISCOVERY

Promoted Stories
- End of an Era: The Car in America
- Trunk
- This app gives
401(k) a
way
- How Absolutely
New York Times
- Bad Neighbor
Warning Sign
- Misperception
Work Environment
- 13 Amazing Uses

More from CNN
- "Miracle material" graphene one step closer
to commercial use
- Astronomers find 'diamond engagement
ring' in space
- Oregon governor performs CPR on woman
- 'Very confident' black boxes detected
- Hyperloop vs. world's fastest trains
- 'Kill switch' may be standard on U.S. phones
in 2015

Imagine an education built toward the right career.
Yours.

University of Phoenix
The Lighthouse

Help people discover content they can trust to be interesting, relevant and timely for them

Distribution Partners

560 MILLION MONTHLY UNIQUE VISITORS GLOBALLY
Over 25 BILLION PAGE VIEWS PER MONTH

Over 200 BILLION RECOMMENDATIONS SERVED PER MONTH
Two Tracks of Personalization

- **Content-based**
  Understand the affinity of each user to each piece of content, based on that user’s historical content consumption

- **Collaborative filtering**
  Understand the affinity of each user to each piece of content, based on co-consumption patterns of the entire user population

Outbrain’s platform is a hybrid that taps both sources
**CONTENT BASED RECOMMENDATIONS**

Standard three step recipe:

1. **Understand** what content is about
   - Represent content in a rich feature space, consisting of site sections, categories, topics, entities

2. **Aggregate** the features of the content consumed by each user to model that user
   - Represent the user in a similar feature space

3. **Match** content to a user

**CONTENT CONSUMPTION EVENTS TURNED INTO PROFILES**

Page Views, Clicks, Shares, Video play/pause, ...

April 17, 2016 236620 Big Data Technologies
### USER PROFILE: FAVORITE CATEGORIES

**YOUR CATEGORIES**

<table>
<thead>
<tr>
<th>Category</th>
<th>Computed</th>
<th>Server Side</th>
<th>Cold Start</th>
<th>Cookie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Football</td>
<td>0.19639465</td>
<td>0.19639464</td>
<td>0.29425267</td>
<td>0.125</td>
</tr>
<tr>
<td>Basketball</td>
<td>0.10662436</td>
<td>0.10662436</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Software</td>
<td>0.061638538</td>
<td>0.061638538</td>
<td>--</td>
<td>0.125</td>
</tr>
<tr>
<td>Real Estate</td>
<td>0.060585096</td>
<td>0.060585093</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

### USER PROFILE: FAVORITE SITES

**YOUR SOURCES**

<table>
<thead>
<tr>
<th>Source</th>
<th>Computed</th>
<th>Server Side</th>
<th>Cold Start</th>
<th>Cookie</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.mtgpj.xi.com/">http://www.mtgpj.xi.com/</a></td>
<td>0.11572189</td>
<td>0.11572186</td>
<td>--</td>
<td>0.00533136</td>
</tr>
<tr>
<td><a href="http://www.icnrental.com/">http://www.icnrental.com/</a></td>
<td>0.096729186</td>
<td>0.09672918</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td><a href="http://www.ynet.co.il/news/3.73464/3.90.html">http://www.ynet.co.il/news/3.73464/3.90.html</a></td>
<td>0.07776443</td>
<td>0.07776443</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td><a href="http://www.ynet.co.il/news/3.73464/2.90.html">http://www.ynet.co.il/news/3.73464/2.90.html</a></td>
<td>0.07048833</td>
<td>0.07048833</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td><a href="http://www.mtgpj.xi.com/">http://www.mtgpj.xi.com/</a></td>
<td>0.06667925</td>
<td>0.06667924</td>
<td>--</td>
<td>0.06666667</td>
</tr>
<tr>
<td><a href="http://www.rickyhl.com/">http://www.rickyhl.com/</a></td>
<td>0.058901973</td>
<td>0.058901973</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td><a href="http://www.mtgpj.xi.com/news/local/">http://www.mtgpj.xi.com/news/local/</a></td>
<td>0.0586125287</td>
<td>0.0586125287</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
USER PROFILE: SITE-CATEGORY COMBINATIONS

YOUR SOURCE-CATEGORY PAIRS

USER PROFILE: FAVORITE TOPICS

YOUR TOPIC MODELS
ILLUSTRATIVE USER TOPICS REPRESENTATION

IMPLEMENTATION (ONLINE AND INCREMENTAL)
Why Online and Incremental?

• Why not update user profiles in batch mode, e.g. nightly/weekly, via Hadoop?
• Users’ session-based access pattern:
  – Have some time, access a publisher, read a few stories, go away for hours
  – Batch updates would incorporate each session’s preferences only after session has long ended
  – Online updates can leverage session’s preferences while session is still in progress
• Better experience for new users
  – Profile can bring value after relatively few interactions, within minutes of user acquisition – why wait a day to delight?

Infrastructure Building Blocks

• Apache Cassandra (née Facebook)
  – Distributed NoSQL database
  – Horizontally scalable through sharding and replication
  – Multi-DC aware via cross-DC replication
• Apache Kafka (née LinkedIn)
  – Scalable publish-subscribe messaging system
  – Messages are consistently partitioned by key
  – Consumers reading a partition consume messages in FIFO order
**UPDATING THE PROFILE – CONCEPTUAL FLOW**

User Interaction
- DocID
- userID
- Action

Categories:
- Nature
- Leisure

Topics:
- Hiking
- Yosemite

UserID’s Profile
- Categories:
  - Football
  - Investing
  - Leisure
- Topics:
  - Hiking
  - Startups
- Entities:
  - Yosemite

Click!

April 17, 2016
236620 Big Data Technologies

**UPDATING THE PROFILE – ARCHITECTURAL FLOW**

Raw Events
- User interaction
- Cassandra

WhoAmI (user service)

Outbrain Gateway
- Get Doc Features
- Write User Event

Document Service

UPAG
- Kafka
- Aggregate & Decay

Online
- Cassandra

Offline
- Cassandra

Features
- Write
- Top-K
- Aggregate
- Produce
- Consume

Event
RAW EVENTS

- In addition to user interests, Outbrain’s user profile also saves some raw events (non-aggregated data)
- Example: several hundred recent pages views per user
- Motivation: do not recommend content the user has already consumed by filtering candidates against this list
  - Also filter some recently or frequently served recommendations

SERVING FLOW AND UPDATE FLOW

- The offline repository stores the full user profile
- The online repository stores only the data required for serving: top-K values of user preference
- Motivation: minimize data to be read at serving, reduce latency
KAFKA PARTITIONING AND CACHING

- Updating a user’s profile requires reading it
- Users consume content in bursts – locality of reference
- By partitioning Kafka messages by userID, all interactions of a user are routed to the same consumer, allowing the consumer to cache user profiles
MULTI-DATACENTER SUPPORT: PROBLEM

- Within a single DC, the usage of Kafka ensures that per user, interactions are processed sequentially in FIFO order by UPAG
- Outbrain operates out of several data centers, and every user’s interactions may be arrive to each one
- Cassandra has cross-DC replication capabilities: a key updated here can (soon) be read there
- Processing interactions independently in each DC might cause race conditions between Cassandra reads and writes across DCs

MULTI-DATACENTER SUPPORT: SOLUTION

- Maintain offline and online profiles per user per DC
- Update flow: each DC updates its own set of profiles with the interactions it processes
  - Those get replicated to other DCs, where they are read-only
- Serving flow: in each DC, WhoAml reads, aggregates and top-k’s all online profiles of the user
- Potential loss of precision in final top-k
SIZE-BOUNDED WEIGHTED SET API

• Each feature type (categories, sites, topics, site-category) contains multiple values per profile, each with a weight
• Some feature types (e.g. entities) have huge value spaces
  – For efficiency and SLA reasons, a profile will not hold more than a bounded number (e.g. several hundred) of those
  – So previously encountered values may be removed from the profile
• Update operations not only change the weights of values present in the current interaction, but may also decay weights of (and remove) values not present in the current interaction

April 17, 2016 236620 Big Data Technologies

SIZE-BOUNDED WEIGHTED SET API

• Profiles (online and offline) are represented as a collection of size-bounded weighted sets (SBWS)
• To Cassandra, a SBWS is just a blob of data
• Supported operations:
  1. Update: given the current SBWS of the offline profile and the (small) SBWS of the current interaction, update the profile’s SBWS
  2. Top-k: given the current SBWS of the offline profile, derive the SBWS of the online profile by a top-K operator
  3. Merge: given several SBWS instances (corresponding to different data centers), merge to a single SBWS
• There can be many implementations, especially for (1)

April 17, 2016 236620 Big Data Technologies
Learning to aggregate

• Users consume sequences of documents
  • Documents are annotated with semantic metadata (categories, topics, entities)
  • How should we transform the activity stream into a profile representing users’ current content affinities and interests?
  • Profile should be incrementally maintained rather than recomputed from all raw events over and over again

Learning to aggregate – proposed methodology

• Difficulties:
  – Explicit consumption data; rejection is implicit
  – Consumption heavily biased by publisher’s editorial staffs
  – Popularity bias is always tricky to overcome in recommender systems

• Methodology:
  – Build user profiles incrementally from training stream of raw events
  – Test procedure: match two users to two stories, based on profiles and story semantics (test cases chosen to have exactly one correct result)
In addition to user interests, Outbrain’s user profile also saves some raw events (non-aggregated data).

Example: several hundred recent pages views per user.

Motivation: do not recommend content the user has already consumed by filtering candidates against this list.

Fact: Outbrain’s crawler assigns document ids sequentially.

- So recent documents have close-by document identifiers.

---

Compression opportunity: since users most often consume current content, most recent page-views have close-by docids.

Solution: **d-gap** encode the docids of recent page views, as done when compressing postings lists.

Twist: gaps can be **negative**.

Can’t sort the gaps, since they are naturally sorted by recency.

Solution: add a sign bit, or encode XOR-gaps.
**ADDITIONAL TOOLING AROUND USER PROFILES**

- Profile support tool: given a profile, what are the historical interactions that justify each property?
  - Relies on the stored raw events
  - Very useful for “reverse engineering” or debugging a profile
- Search tool: index all profiles by a search engine, for the ability to query them
  - Find me all CNN users who are interested in home improvement and golf
  - Given a user, find similar users
- Profiles dump for offline analytics: have all profiles on a DFS for offline analytics and modeling (e.g. via Hadoop, Spark, etc.)
  - Recall that serving use case only supports access by uuid
  - Need to maintain freshness of dump as users interact with more content

**SOME NUMBERS ABOUT SCALE**

- 3 Data centers
- Accommodates 35,000 recommendation requests per second, served on average at 50 ms
- 30 WhoAmI machines
- 8 UPAG machines
- 2 Cassandra clusters of 80 + 20 machines with 30 TB of compressed data
- 300,000/sec read operations from Cassandra at < 10 ms per operation