Wolfram Wingerath

Big Data Analytics
With AWS Athena

Big Data
Big Data Analytics With AWS Athena

Wolfram Wingerath, Code.Talks 2019
Research:
• Stream Processing
• Real-Time Databases
• NoSQL & Cloud Systems
• ...

Practice:
• Web Caching
• Big Data Analytics
• Anger Management
• ...

I Am Wolle
I Like Real-Time Stuff

2018

Real-Time Processing Explained
A Survey of Storm, Samza, Spark & Flink

2017

Real-Time Databases Explained
Why Meteor, RethinkDB, Parse & Firebase Don't Scale

latency
throughput
Let’s Talk About Batch Analytics

Collection

- Raw PI tracking & meta data
- Custom tracking

Ingestion

- Tracking (RUM)

Analytics

- Materialized views & aggregations
- Historical data

Reporting

- Performance Dashboard
- QA Dashboard
- Real-Time Alerting
- Ad-hoc SQL Interface
- Custom Reporting
What's in the Data?
3 Things Make Your Website Slow

1. Backend Processing
2. Network Delays
3. Client
We Make Websites Fast

Website + Speed Kit (Service Worker)

Kino 7
Friday, 16:00

Fast Requests

Speed Kit Cloud

Real-Time Sync

Origin Server

Tracking & Ad Services
Accelerating Personalized HTML

Browser

Merge

Fast & Anonymous

Speed Kit Cloud

Origin Server

Slow & Personalized
Split Testing for Web Performance

Speed Kit Users
- Speed Kit enabled

Tracking

Measurable uplift:
- Performance
- User engagement
- Business success

Normal Users
- Speed Kit disabled (no acceleration)

Tracking

Kino 7 Friday, 11:00
Goal: Performance & Business Insights

- Time-to-First-Byte
- First (Contentful) Paint
- DOM Timer
- First Input Delay

- Session Length
- Time on Site
- First User Interaction
- Bounce Rate

- Cart Size
- Transactions
- Conversion Rate
- Revenue

- Page Views & Sessions
- Browser Distribution
- JavaScript Errors
- Caching Insights

Performance
User Engagement
Business KPIs
QA Metadata

- Timing API
- Service Worker
- Unhandled Errors

Browser → Cloud Backend → Tracking Beacon
How to Monitor Performance?

- Logging requests is not enough:
  - User? Rendering? ...
  - Browser cache (invisible)
  - Origin requests (no logs)
  - CDN requests

- Solution: Tracking every PI (page impression)
Types of Data Beacons

1. 1 for **static** info
   (URL, user agent, session ID, ...)

2. 1 for **timings**
   (TTFB, load time, FCP, ...)

3. 0–n for **custom** events
   (first input, add-to-cart, ...)

**Metric Beacon**

**CustomEvent Beacon**
Let's Analyze the Data
### Tracking Data in MongoDB

- **Metric**
  - URL
  - User agent
  - User ID
  - Session ID
  - Navigation start
  - Load
  - Split test group
  - ...

- **CustomEvent**
  - Event type
  - Value

- Debugging with OLTP **queries**
- Analyses with **aggregation pipeline**, e.g.:
  - Average session length
  - Uplift vs. Acceleration (example: Conversion rate vs. first paint uplift)
  - Different browser timings by device/browser/...
CDN Logs in S3

- Automation via Jenkins jobs
- **Caching-related** statistics, e.g.:
  - Request latency distribution (histogram)
  - Image optimization efficiency
Problems I: CDN Data Import

1. **Partitioning** by hour, but not by customer
   - Not scalable!

2. **Indexing** & database import:
   a) Import into indexed table
   b) Drop index → import → create index  
      each takes forever
   c) Query table without index
Problems II: Aggregation Pipeline

- **Indexing**
  Queries over non-indexed attributes were infeasible

- **Scalability**
  Queries got slower with increasing amounts of data

- **Runtime**
  Even with indexes in place, queries could take 30+ min.

- **Complexity**
  MongoDB aggregation pipelines become sophisticated quickly
III: Reporting

- Scheduling:
  1. On-demand
  2. Periodic job

- Problems:
  - Cumbersome to build & maintain
  - Awkward to extend
  - Unreliable

Automated reports, Nice diagrams, New problem
Problems IV: Joins in MongoDB

Joins infeasible for data-intensive queries!

- No conversion analysis
- No business uplift validation
- Not acceptable!
Fixing My Life With Flex Tape Athena
The „A“ Stands for „AWSome“

- Desperate attempt:
  1. Dump MongoDB collection
  2. Upload to S3
  3. Query with Athena

- Typical analysis:
  - 1 equi-join
  - 3 mio. Pls
  - ~15+ min.
The „A“ Stands for „AWSome“

- Desperate attempt: New best practice:
  1. Dump MongoDB collection
  2. Upload to S3
  3. Query with Athena

- Typical analysis:
  - 1 equi-join
  - 3 mio. Pls
  - ~10 seconds
What's an Athena?

- **Managed Presto:**
  - Interactive analytics with SQL
  - Heterogeneous datastores
  - Petabyte-scale (Facebook)

- **Pricing** by scanned data volume:
  - Efficient storage formats!
  - Partitioning or clustering!
  - Careful query design!

---

*Raghav Sethi, Martin Traverso, Dain Sundstrom, David Phillips et al., *Presto: SQL on Everything*, ICDE 2019*
Upgrading Our ETL Pipeline

- **Simplicity:**
  - Everything in one place
  - Easy to access (SQL)

- **Scalability & efficiency:**
  - Hundreds of gigabytes scanned in a query
  - Response time on the order of seconds
Processing Stages & Latency

**Alerting**
- Simple metrics
  - Counters
  - Extreme values
  - Specific errors

**Processing Stages**

**Trend Analysis**
- Complex aggregations
  - Conversion rate
  - Performance by day
  - Seasonal effects
### Stage 0: Data Preparation

```json
{
    "_id": "ABC",
    "loadEvent": {
        "$numberLong": "1571101211368"
    },
    "createdAt": {
        "$date": "2019-10-15T01:00:11.462Z"
    },
    ...  
}
```

- **Schema Definition**
  1. **Tables** for raw data
  2. **Views** on top to hide artifacts
Example: Timestamps

FROM_UNIXTIME(
    CAST(
        CAST(
            JSON_EXTRACT(loadEventEndRaw, '$"$numberlong"')
        AS VARCHAR)
    AS DECIMAL (38,3)) / 1000
) AS loadEventEnd

1. Extract UNIX timestamp from JSON
2. Cast to varchar
3. Cast to decimal
4. Divide by 1000
5. Convert to timestamp
CASE WHEN
    CAST(
        CAST(
            JSON_EXTRACT(loadEventEndRaw, '"$numberlong"')
        AS VARCHAR)
    AS DECIMAL (38,3)) > 0
THEN
    FROM_UNIXTIME(
        CAST(
            CAST(
                JSON_EXTRACT(loadEventEndRaw, '"$numberlong"')
            AS VARCHAR)
        AS DECIMAL (38,3)) / 1000
    )
END AS loadEventEnd

1. Extract UNIX timestamp from JSON
2. Cast to varchar
3. Cast to decimal
4. Divide by 1000
5. Filter out rubbish
6. Convert to timestamp

Note: still a timestamp (no timing in ms)
Stage 1: Join Beacons

- Consolidate PI data into single rows
- Data cleaning (e.g. nullify when `loadEnd < navigationStart`)
Stage 2: Resolve User Agents

- **Metric_complete**
  - User agent
  - ...

- **Metric_complete**
  - browser (+ version)
  - Device type
  - Device
  - OS (+ version)
  - ...

- **Paid service**: interpreting user agents is complex!
- **Fallback**: simple case-when logic for browsers and device type
- **Simplification** required, e.g.:
  - Device: mobile/desktop/tablet/server/game-console/wearable/vehicle/...
  - Browser: Chrome/Firefox/Safari/Opera/IE/Netscape/Tesla-Browser/...
Stage 3: Session Metrics

- **Aggregation** by session ID
  - Session length, bounces
  - Time on site
  - Performance (e.g. median FCP)
  - Conversions

- Count as conversion? 
  - Yes
  - No (Checkout (reload))
Stage 4: Materialized Views

- Business uplift during Proof of Concept (PoC)
- Aggregation over days or weeks
- Performance and business trend analysis
Reporting: The Right Tool for the Job

Custom Code → Unified analytics

- Automated reports
- Nice diagrams
- New problem
Reporting: The Right Tool for the Job

- **Requirements:**
  - Automation
  - Easy data exploration
  - Robustness
Why QuickSight?

- Easy integration with Athena
- Ease-of-use
- Quick results

AWS Services:
- S3
- Athena
- QuickSight

Integration tools:
- fastly
- Baqend
Worst Practices
Speed Kit PoC Analysis Dashboard

Executive Summary of Uplifts (Positive Value = Improvement)

<table>
<thead>
<tr>
<th>skgroup</th>
<th>Last X Days (Including Yesterday)</th>
<th>Device Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKHTML</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>all</td>
</tr>
<tr>
<td>Sessions</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Orders</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Conversations</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Order Value</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>skewed Order...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Order Uplift</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>skewed Order...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Order Value Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Load Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>First Paint Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Session Length Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Return Rate Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Bounces Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>ADV Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td># Fit Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td># Session Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td># User Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>addToCart Uplift...</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td># addToCart Uplift...</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Why, Oh, Why, QuickSight?
Time Works Differently in AWS Dashboards

Running Query...

Estimated time elapsed: 0 seconds

You can run another query by clicking on the New Query button. The current query will continue to run in the background. You can check the status of all queries in the History Tab.
Time Works Differently in AWS Dashboards

Congratulations! Your query took 2 seconds only!
Click Once, Pay Twice

1. Data Set 53.6KB
   - Import complete: 100% success
   - 580 rows were imported to SPICE
   - 0 rows were skipped

2. Last refreshed: 20 hours ago
   - Refresh Now
   - Schedule refresh

3. View summary

4. Note: You haven’t updated the data set, yet.

   /* Quicksight c26063a5-b35a-4682-a6d8-e5e7f0a1ea1f */
   SELECT "app",
   "days_in_past",
   "responsecause",
   ...
But There is **More**!

- **Questionable Limits**
  Diagrams in the dashboard, query timeouts, etc.

- **Exhausted Resources** in Athena
  Queries fail depending on time, day, and weather

- **Implicit Query Rewriting** in QuickSight
  Hard to predict, hard to circumvent, hardly helpful
• Next Steps:
  • Continuous ingestion
  • Streaming analytics & real-time dashboards
  • Drop QuickSight
Real-Time Analytics & Reporting

Next Steps:
- Continuous ingestion
- Streaming analytics & real-time records
- Drop QuickSight
Thanks! Any Questions?

- Kino 3 Thursday, 16:00: GDPR Panel Discussion
- Kino 7 Friday, 11:00: Web Performance Talk
- Kino 7 Friday, 16:00: Service Worker Hackathon

Wolfram Wingerath, Data Engineering  ww@baqend.com  Baqend
Thanks! Any Questions?

- GDPR Panel Discussion
  - Right Here
  - Right Now

- Web Performance Talk
  - Kino 7
  - Friday, 11:00

- Service Worker Hackathon
  - Kino 7
  - Friday, 16:00

Wolfram Wingerath, Data Engineering
ww@baqend.com
Baqend