CLICKSTREAM ANALYSIS WITH APACHE SPARK

Andreas Zitzelsberger
THE CHALLENGE
ONE POT TO RULE THEM ALL

Web Tracking
- Clicks & Views
- Conversions

ERP
- Products
- Inventory
- Margins

Ad Tracking
- Ad Impressions
- Ad Costs

CRM
- Customer
- Orders
- Creditworthiness

Clicks & Views
Conversions
Ad Impressions
Ad Costs
Customer
Orders
Creditworthiness
ONE POT TO RULE THEM ALL

Monetarization

Retention  Reach

steer …
- Campaigns
- Offers
- Contents
REACT ON WEB SITE
TRAFFIC IN REAL TIME
SAMPLE RESULTS

- Geolocated and gender-specific conversions.
- Frequency of visits
- Performance of an ad campaign
THE CONCEPTS
THE FIRST SKETCH

JSON (4000/sec)

Magic

15min (= real-time)

SQL

Qlikview (BI)

R (Data Science)

pig (Reporting)
Magic
CALCULATING USER JOURNEYS

KPIs:
- Unique users
- Conversions
- Ad costs / conversion value
- ...

Event stream:
- C
- V
- V
- C
- V
- V
- V
- VT
- VT
- V
- V
- V
- V
- VT
- C
- V
- X

User journeys:
- C
- V
- V
- V
- V
- V
- V
- V
- V
- C
- V
- VT
- VT
- VT
- C
- X

Web / Ad tracking
THE ARCHITECTURE

Welcome to WONDERLAND

Big Data
„LARRY & FRIENDS“ ARCHITECTURE

Runs not well for more than 1 TB data in terms of ingestion speed, query time and optimization efforts
Nope.
Sorry, no Big Data.
HADOOP & FRIENDS ARCHITECTURE

- **JSON Stream** → **Collector** → **Event Data Lake** → **Batch Processor** → **Analytics DB**

**Steps:**
- **Kafka**
- **HDFS**
- **Hadoop**
- **Hive**

**Issues:**
- Aggregation takes too long
- Cumbersome programming model (can be solved with pig, cascading et al.)
- Not interactive enough
Nope.
Too sluggish.
K-ARCHITECTURE

Cumbersome programming model

Over-engineered: We only need 15min real-time ;-)

Stateful aggregations (unique x, conversions) require a separate DB with high throughput and fast aggregations & lookups.
A-ARCHITECTURE

- Cumbersome programming model
- Complex architecture
- Redundant logic
FEELS OVER-ENGINEERED...

http://www.brainlazy.com/article/random-nonsense/over-engineered
The Final Architecture*

*) Maybe called μ-architecture one day ;-)
FUNCTIONAL ARCHITECTURE

- Fault tolerant message handling
- Event handling: Apply schema, time-partitioning, De-dup, sanity checks, pre-aggregation, filtering, fraud detection
- Tolerates delayed events
- High throughput, moderate latency (~ 1min)

Collection → Raw Event Stream → Ingestion → Atomic Event Frames → Data Lake

- Buffers load peeks
- Ensures message delivery (fire & forget for client)
- Eternal memory for all events (even strange ones)
- One schema per event type. Time partitioned.

Processing

- Create user journeys and unique user sets
- Enrich dimensions
- Aggregate events to KPIs
- Ability to replay for schema evolution

Fact Entries → Analytics Warehouse

- The representation of truth
- Multidimensional data model
- Interactive queries for actions in realtime and data exploration
SERIAL CONNECTION OF STREAMING AND BATCHING

- Obvious choice for cloud-scale messaging
- Way the best throughput and scalability of all evaluated alternatives

Collection → Kafka→ Raw Event Stream → Ingestion → Spark Streaming → Atomic Event Frames → Event Data Lake → Processing → HDFS + Parquet

- Cool programming model
- Uniform dev&ops
- Simple solution
- High compression ratio due to column-oriented storage
- High scan speed

Processing → Event Data Lake → HDFS + Parquet → Spark → SQL Interface

- Cool programming model
- Uniform dev&ops
- High performance
- Interface to R out-of-the-box
- Useful libs: MLlib, GraphX, NLP, …

SQL Interface → Analytics Warehouse

- Good connectivity (JDBC, ODBC, …)
- Interactive queries
- Uniform ops
- Can easily be replaced due to Hive Metastore
public Map<Long, UserJourney> 
sessionize(JavaRDD<AtomicEvent> events) {

return events
  // Convert to a pair RDD with the userId as key
  .mapToPair(e -> new Tuple2<>(e.getUserId(), e))
  // Build user journeys
  .<UserJourneyAcc>combineByKey(
    UserJourneyAcc::create,
    UserJourneyAcc::add,
    UserJourneyAcc::combine)
  // Convert to a Java map
  .collectAsMap();
}
STREAM VERSUS BATCH
APACHE FLINK

- Also has a nice, Spark-like API
- Promises similar or better performance than spark
- Looks like the best solution for a κ-Architecture

- But it’s also the newest kid on the block
EVENT VERSUS PROCESSING TIME

- There’s a difference between even time ($t_e$) and processing time ($t_p$).
- Events arrive out-of-order even during normal operation.
- Events may arrive arbitrary late.

- Allow arbitrary update windows of metrics.
- Apply a grace period before processing events.
Resolution in Time

- Year
- Quarter
- Month
- Week
- Day
- Hour
- Minute

- $t_i$: Ingestion time
- $t_e$: Event Time
- $dt_p$: Aggregation time frame
- $dt_w$: Grace period

- $t_p$: Processing Time
- $i$: Insert fact
- $u$: Update fact
BEST-OF-BREED INSTEAD OF COMMODITY SOLUTIONS

Polyglot Processing

ETL
Analytics
Realtime Analytics
Slice & Dice
Data Exploration

http://datadventures.ghost.io/2014/07/06/polyglot-processing
NO RETENTION PARANOIA

Data Lake

- Eternal memory
- Close to raw events
- Allows replays and refills into warehouse

Strange Events

Analytics Warehouse

Aggressive forgetting with clearly defined retention policy per aggregation level like:
- 15min:30d
- 1h:4m
- ...
BEWARE OF THE HIPSTERS
ENSURE YOUR SOFTWARE RUNS LOCALLY

The entire architecture must be able to run locally. Keep the round trips low for development and testing.

Throughput and reaction times need to be monitored continuously. Tune your software and the underlying frameworks as needed.
TUNE CONTINUOUSLY

Load generator

Collection ➔ Raw Event Stream ➔ Ingestion ➔ Atomic Event Frames ➔ Event Data Lake ➔ Processing ➔ Fact Entries ➔ SQL Interface

Throughput & latency probes

System, container and process monitoring
Overall dev effort until the first release: **250 person days**

Dimensions: **10**  
Integrated 3rd party systems: **7**

KPIs: **26**

Inbound data volume per day: **80GB**

New data in DWH per day: **2GB**

Total price of cheapest cluster which is able to handle production load:
Your own Datacenter (less than 500$)

- SPARK Master
- SPARK Worker
- ZOOKEEPER
- SHARD #0-3
- SOLR Cloud

- Odroid XU4
  - 2 GB RAM
  - 64 GB eMMC Disk
  - Ubuntu Linux
  - $70

- SOLR Cloud
  - SHARD #4-7
  - SPARK Worker

- SOLR Cloud
  - SHARD #8-11
  - SPARK Worker

- SOLR Cloud
  - SHARD #12-15
  - SPARK Worker

- SOLR Cloud
  - SHARD #16-19
  - SPARK Worker

Odroid Hardkernel

- Odroid XU4
  - 2 GB RAM
  - 64 GB eMMC Disk
  - Ubuntu Linux
  - $70

40 Cores
10 GB RAM
320 GB total disk
THANK YOU

@andreasz82

andreas.zitzelsberger@qaware.de
BONUS SLIDES
CALCULATING UNIQUE USERS

- We need an exact unique user count.
- If you can, you should use an approximation such as HyperLogLog.

CHARTING TECHNOLOGY

https://github.com/qaware/big-data-landscape
CHOOSING WHERE TO AGGREGATE

1. Ingestion
- Enrichment
- Preprocessing
- Validation

2. Processing
- The hard lifting.

3. Analytics
- Processing steps that can
  be done at query time.
- Interactive queries.

- Enrichment
- Preprocessing
- Validation

- Event Data Lake

- Analytics Warehouse

- Fact Entries

- Atomic Event Frames

- Processing steps that can be done at query time.
- Interactive queries.