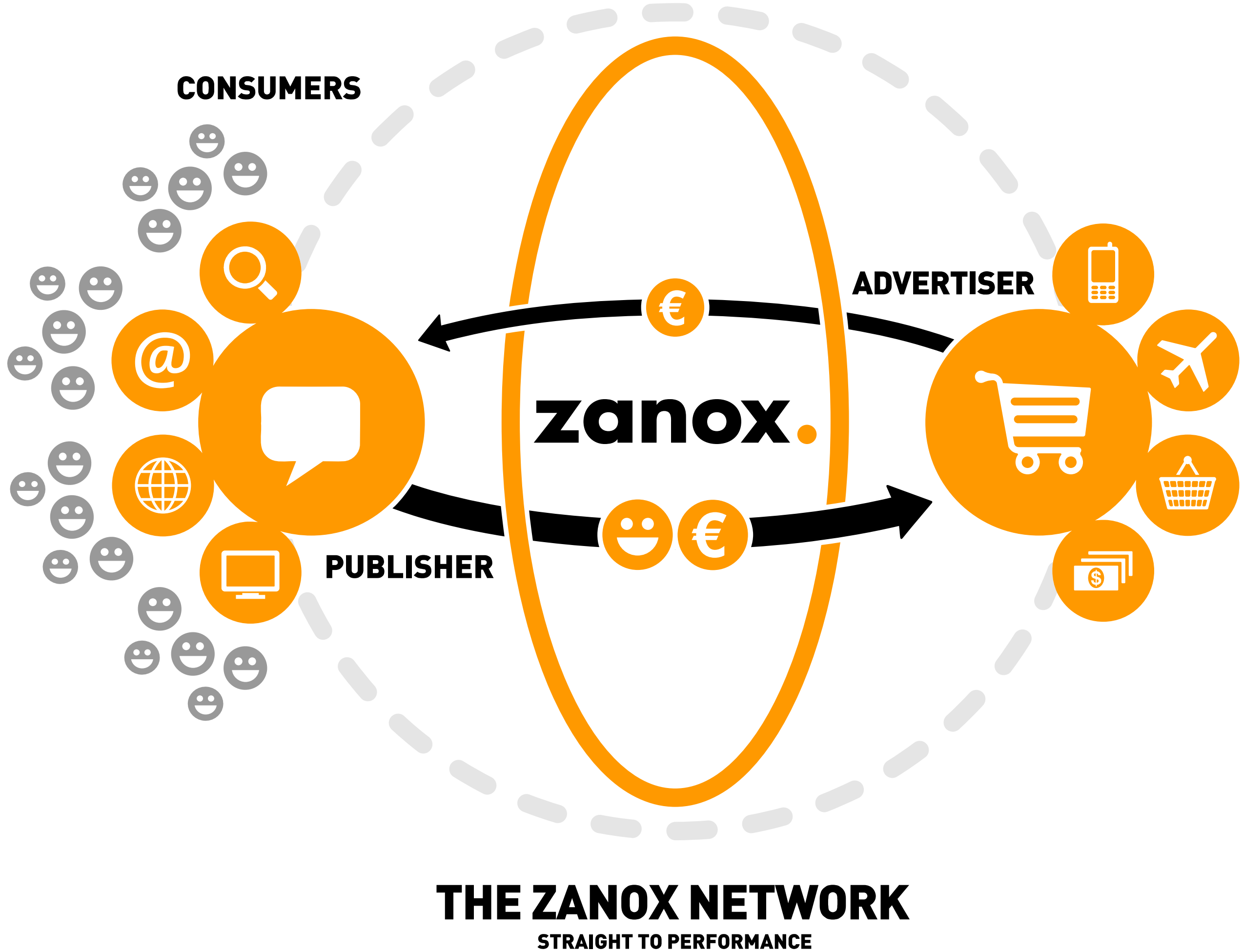
DENORMALIZING DATA SETS IN HADOOP WHILE BUILDING LUCENE INDEXES

Zürich | 06.09.13

TABLE OF CONTENTS

THE ZANOX NETWORK



WHO AM I?

- Senior Architect at Zanox
  - 7+ years experience in building distributed search systems based on Lucene
    Lucid Certified Apache Solr/Lucene Developer
  - 5+ years of using Hadoop and 2+ years of using HBase for data mining
    Cloudera Certified Developer for Apache Hadoop and HBase
- I have applied different machine-learning techniques mainly to optimise resource usage while performing distributed search during my PhD
  - See my book: "Beyond Centralised Search Engines
    An Agent-Based Filtering Framework"
- How can you reach me?
  - dragan.milosevic@zanox.com
  - http://www.linkedin.com/in/draganmilosevic

1. ABOUT ZANOX AND ME
2. **REPORTING IN GENERAL**
3. MAP & REDUCE SIDE JOINS
4. QUESTIONS

REPORTING IN GENERAL

[illegible]:00 00:[illegible] 00:00:00.[illegible]

Data-Import Data producers are pushing data during a whole day

MR Jobs Reporting jobs are started at midnight
It takes around **6 hours to process data** from yesterday
Index loading normally lasts no more that 30 minutes

Reporting There are **multiple search clusters** for redundancy
Every cluster has one merger that coordinates searchers
The health of every cluster is ensured by observer
Searchers are grouped based on their capabilities
Within 100ms queries are propagated to best searchers
Capable to provide reports in **sub-second time**

1. ABOUT ZANOX AND ME
2. REPORTING IN GENERAL
3. **MAP & REDUCE SIDE JOINS**
4. QUESTIONS

MAP & REDUCE SIDE JOINS

Map-Side → Reduce-Side → Both-Side

Advantages

- **Reducers are not needed** for joining
- MR pipeline can be used for other purposes
- Extremely friendly regarding network
- **Distributed cache** supports distribution of dataset that should fit in memory
- Save smaller dataset with higher HDFS replication as alternative to distributed cache
- **JVM reusing** minimises starting of map-tasks

Disadvantages

- One dataset has to be small to fit into memory
- If only map tasks are used, potentially produces many small files (one per map-task)

MAP & REDUCE SIDE JOINS

Map-Side → Reduce-Side → **Both-Side**

Advantages

- Datasets of any size can be joined
- Easy to implement because MP helps

Disadvantages

- **Loads significantly network** because both datasets need to be completely transferred to reducers
- Keys that mappers generate need to be tagged with source information to support **secondary sorting**
- **Skew** in the distribution of a join key imposes significant challenges to load uniformly available reducers

zanox.

MAP & REDUCE SIDE JOINS

Map-Side → Reduce-Side → Both-Side

Advantages

Frequent join keys are handled on map-side
Reducers are needed only for not frequent join keys
Consequently it is **extremely network friendly**
Skew in join keys is fixed because infrequent keys cannot make significant imbalances on reducers
Tagging keys that mappers generate with data source is also no more needed because buffering on reducer side becomes again an option

Disadvantages

Many records from the smaller dataset are unnecessarily transmitted to reducers. Bloom filters can help here

1. ABOUT ZANOX AND ME
2. REPORTING IN GENERAL
3. MAP & REDUCE SIDE JOINS
4. **QUESTIONS**

QUESTIONS

?