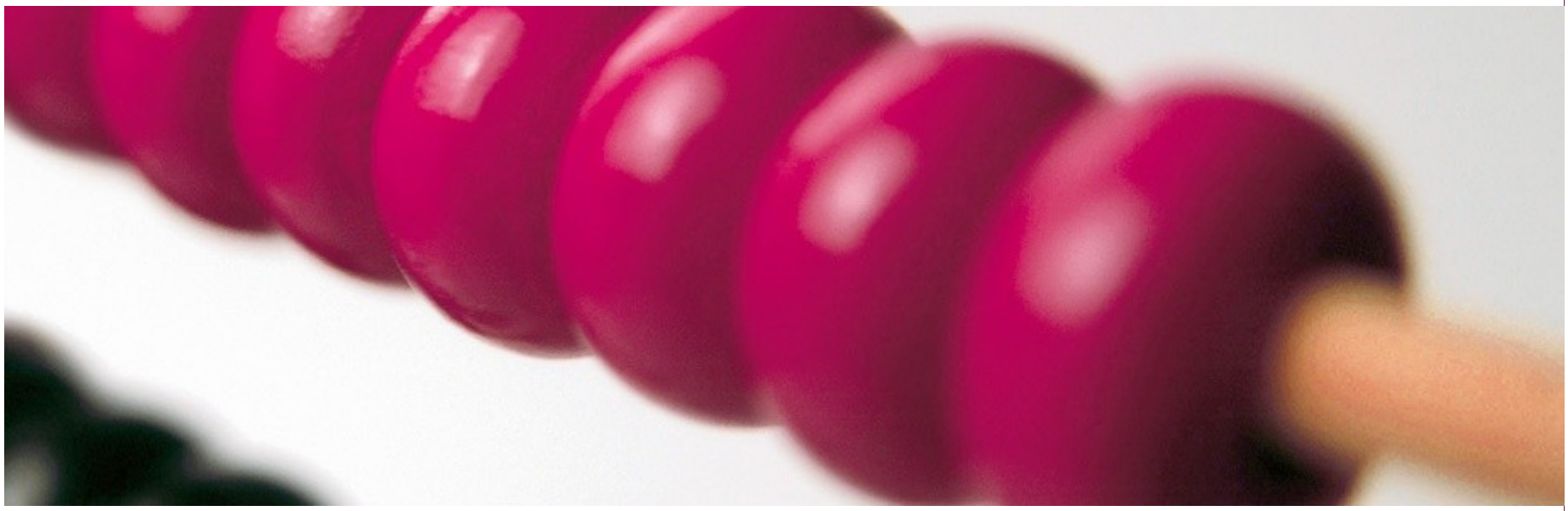
M/R for MR

-



nuRAGO

Who is nurago?

- founded in early 2007
- technology for usability and ad efficiancy research
- part of USYS Nutzenforschung: Hanover, Hamburg, Berlin, Munich, London
- consultants, developers, operation engineers (about 30+ employees)
- driven by technology and methodology

- research platform I: LEOtrace
  - usability research, audiance measurement
  - Proxy Servers, Browser Add-ons
  - sampled data from panel members (2k – 25k UU per project, about 6TB per month overall)

- research platform II: BrandSpector
  - ad efficiancy research
  - Cookies, Online Surveys
  - full data based on media volume (10m – 100m AIs per project, about 500GB per month overall)

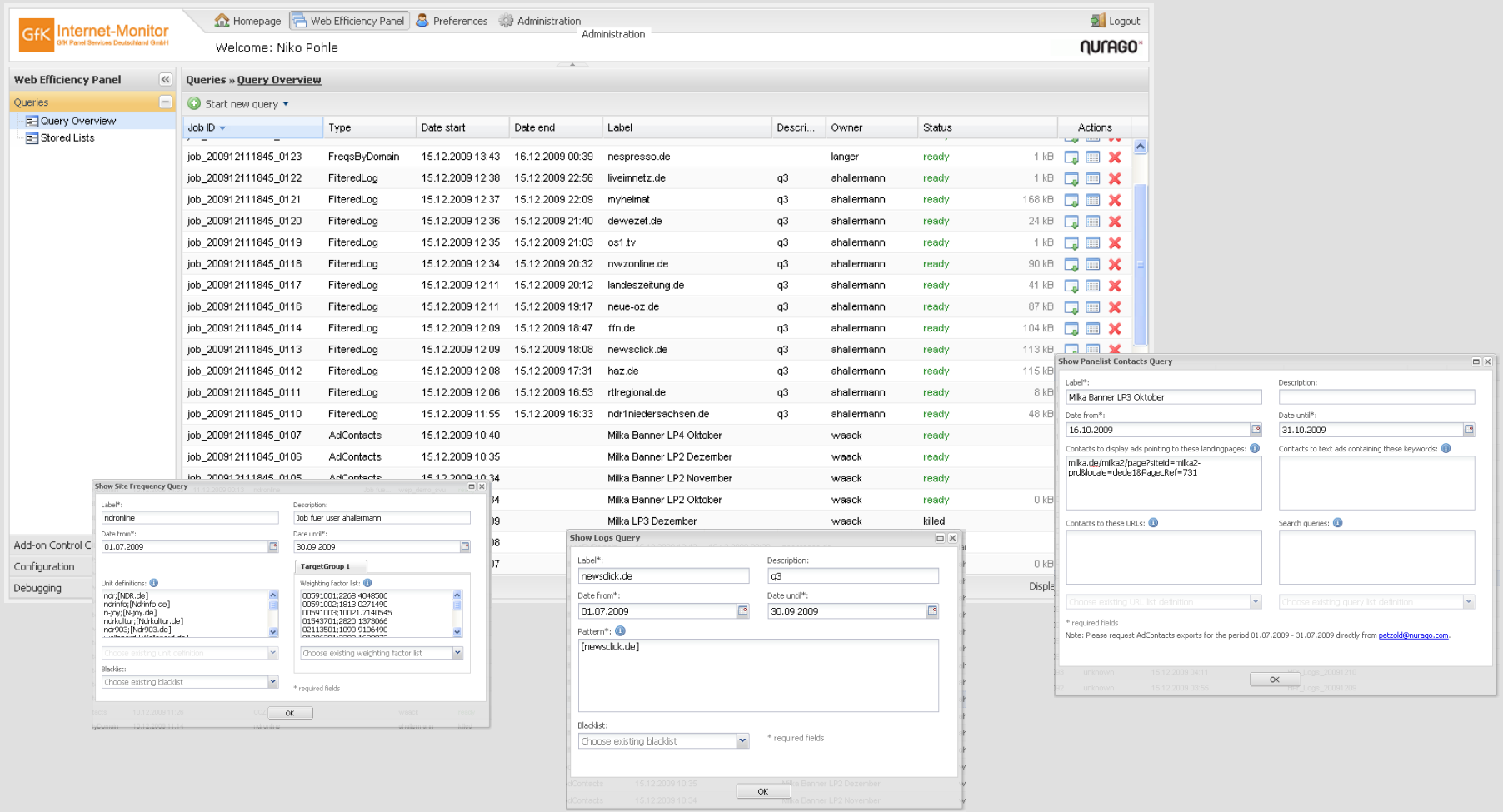
Technology Stack Platform Strategy

- Data Collection Framework
  - LEOtrace Browser Add-Ons provide a unified JavaScript Runtime Environment for IE und FF
    - think of Greasemonkey with a remote control
    - Services API supports study setups (event triggered surveys, DOM manipulation etc.)
  - BrandSpector
    - Unified Tracking Tags embedded into ad creatives or sites

- Data Processing Framework
  - *Magic Happens Here*

- Data Reporting Framework
  - „Portal Server“ providing SSO, I18N, ACLs and pluggable reporting modules
  - PHP Zend Framework for serverside MVC
  - GUI library extJS for consistent Look & Feel plus Charting

LEOtrace Data Processing



Data Processing Framework

- Examples of LEOtrace Data Analysis

  - Input Log Data: UserID, Timestamp, URL, Body, etc.
  - Input Ad Contact: UserID, Timestamp, Landingpage,

  - Standard Output : Frequency By Domains:
    - count the numbers of Page Impressions and Unique Users for certain URL patterns
    - projected gross reach, net reach, sessions, duration per URL pattern
    - *whats is the combined reach of facebook.com and studivz.net?*

  - Standard Output : Filtered Logs
    - advanced grep on the URL based on RegEx
    - UserID, Timestamp, URL, Duration, Session ID
    - *export all males of age 25 to 35 who entered the checkout process on spreadshirt.de*

  - Standard Output : Ad Contacts
    - Advanced grep on Ad Landing Page URL, Google Search Terms, visited URLs
    - UserID, Timestamp, ContactType, ContactDetails

Technology Stack
The Missing Link

nurago
APPLIED RESEARCH TECHNOLOGIES

- Data Transport
  - from web servers to processing nodes: a bunch of shell scripts using scp, rsync, etc.
- LEOtrace: Data Processing ~~Framework~~ Attempts
  - Generation 1: one project = one MySQL DB + Ad Hoc SQL Queries
  - Generation 2: one project = 23 partitioned MySQL DBs + stand alone java apps
  - Generation 2.5: one project = 23 partitioned MySQL DBs + MapReduce Jobs
  - Generation 2.8: one project = 23 partitioned MySQL DBs + HDFS Flat Files + MapReduce Jobs
  - Generation 3: one project = Flat Files in HDFS + MapReduce Jobs
- BrandSpector: Data Processing ~~Framework~~ Attempts
  - Generation 1: one project = one MySQL DB + Ad Hoc SQL Queries
  - Generation 2: one project = Flat Files + awk jobs
  - Generation 3: one project = Flat Files in HDFS + MapReduce Jobs (t.b.d.)

LEOtrace Data Processing

- raw data (XML encoded „events“) is created by Browser Add-ons
- web servers receive and parse the XML into an intermediate format
- data is transfered to one of several processing nodes using rsync
- JAVA application is preprocessing the data and stores results in sharded DBs
- a daily M/R job dumps aggregated data into Flat Files in HDFS (one per day)

- end users issue new queries using a Click&Point GUI based on PHP und extJS
- PHP talks to a Servlet (REST, JSON) who talks to the Hadoop JobTracker
- Job results are written to HDFS and accessed through the Servlet

Data Processing Weaknesses

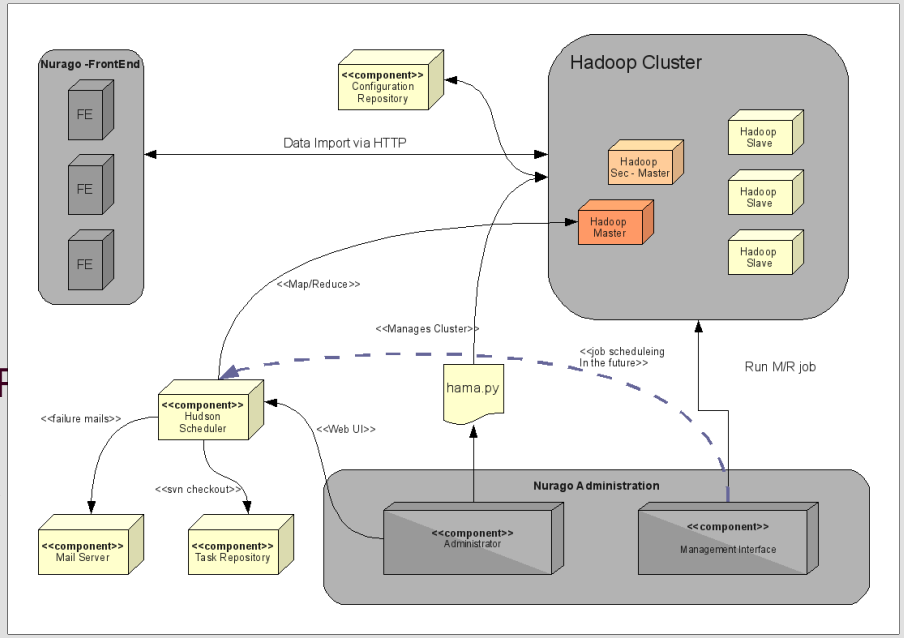
- It does its job, but:
  - still very fragmented storage: many potential pitfalls
  - too much code dealing with the infrastructure (focus more on business logic)
  - stability issues due to M/R jobs being to complex or bloated
  - Job Tracker sometime just freezes with many pending maps (forces a restart of mapred)
  - hard to maintain a set of shared job implementations for common cases
  - Sometimes veeeeery slow (not a real problem but expensive)
  - Troubled Operation Engineers

- we need to get better to meet the demand of internal and external users
  - we need to cover just a small set of rather simple aggregations: counts, distinct counts, greps
  - analysts should be able to issue such queries using a simple GUI
  - there should exist an easy way to implement additional customized jobs and run them in a controlled environment enforcing best practices and reuse code

  - We need one more Framework to complete the Platform Strategy

- **Dudong**: provides a unified framework for data handling and processing

- simplified Cluster Management
  - include/exclude nodes, debian packages
  - centralized description of topology
- unified storage management
  - direct import of data from web servers
  - provides a common data representation
- Dudong Job API
  - supports dynamic COUNT, GROUP, GREP
  - supports Job Chaining
  - covers LEOtrace as well as BrandSpector
- Scheduling
  - based on a Hudson



Look Out

- migrate existing data processing infrastructure
- we are going to scale out to support more international business
  - one unified cluster or separated clusters?
  - how do we charge the cluster utilization?
- more advanced studies means more advanced analytics
  - Taxonomy, Classification, etc.
  - Annotation based on collected content (analyze bread crumbs, shopping baskets, etc.)
- Even more data
  - additional data collection methods are in the pipeline

www.nurago.com

nurago GmbH | applied research technologies | Kurt-Schumacher-Straße 24 | 30159 Hannover
Tel. +49 511 213 866-0 | Fax +49 511 213 866-22

Contact

- Nikolaus Tecklenburg

nurago GmbH
applied research technologies
Kurt-Schumacher-Straße 24
30159 Hannover

Backup

- unsere Input und Output Daten sind tabellenbasiert..
- Dudong erlaubt uns dynamisch auf Basis des Spaltenindex (Nummer der Spalte) Records und Keys zu erstellen
- Beispiel: Spalte 1 soll als Long behandelt werden, Spalte 2 soll als Text behandelt werden..
- Dadurch können ggf. in Zukunft weniger-komplexe Aufgaben schneller gelöst werden
- Die JobChain erlaubt es mehrere Jobs einfacher hintereinander zu hängen. dabei können für die temporären Outputs Policies mitgegeben werden, die zum Beispiel dafür sorgen, dass temporäre Dateien erst gelöscht werden wenn spätere Jobs erfolgreich waren
- zusätzlich können mehrere JobChains ineinander kaskadiert werden
- ListRecords und CombinedKeys können durch das oben angesprochene RecordSchema (Index -> Hadoop-Datentyp) erzeugt werden
- Allgemeine Counterjobs (Zählung von Records) bzw. Filterjobs (Filterung von Records) können allgemeine Aggregationen übernehmen
- Über ExecutionCallbacks kann auf Zustände eines asynchronen Jobs reagiert werden