Data-driven applications in practice

2017-05-11

Lars Albertsson
www.mapflat.com

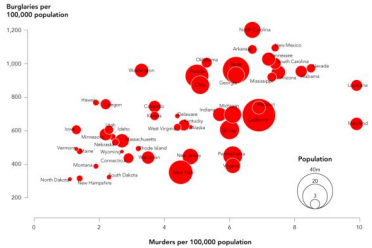
Who’s talking?

- KTH-PDC (MSc thesis)
- Swedish Institute of Computer Science (distributed system test+debug tools)
- Sun Microsystems (very large machines)
- Google (Hangouts, productivity)
- Recorded Future (natural language processing startup)
- Cinnober Financial Tech. (trading systems)
- Spotify (data processing & modelling)
- Schibsted Products & Tech (data processing & modelling)
- Mapflat - independent data engineering consultant

Most companies have

User behaviour

Business insights



Many dream of

User behaviour

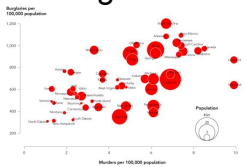
External sources

Experiments

Curated content

Exploration

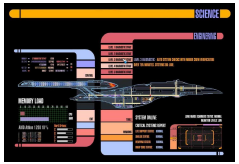
Business intelligence



User content

Professional content

Systems

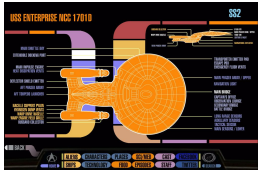


Do Big Data things

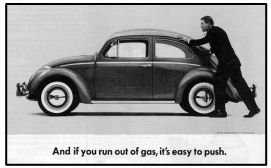
Data-based features

Recommendations

System diagnostics

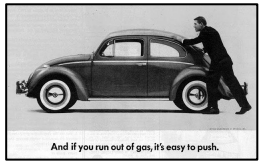


Ads / partners

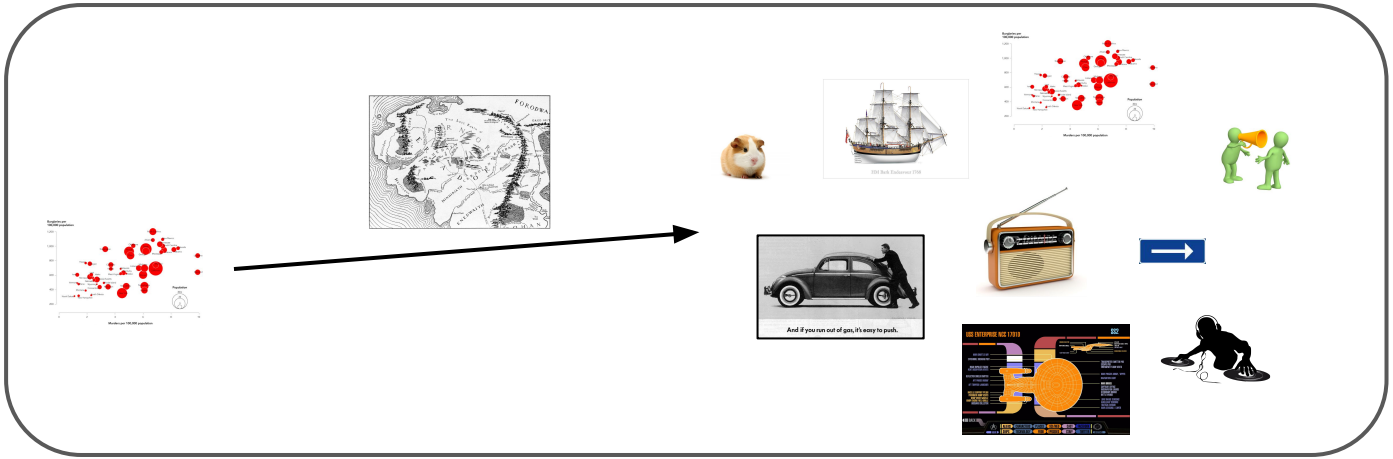


Pushing

Ads



Presentation objective



Data & AI will affect everyone

Steam engines

Cars

Computers / digitalisation

Internet

Smartphones

AI

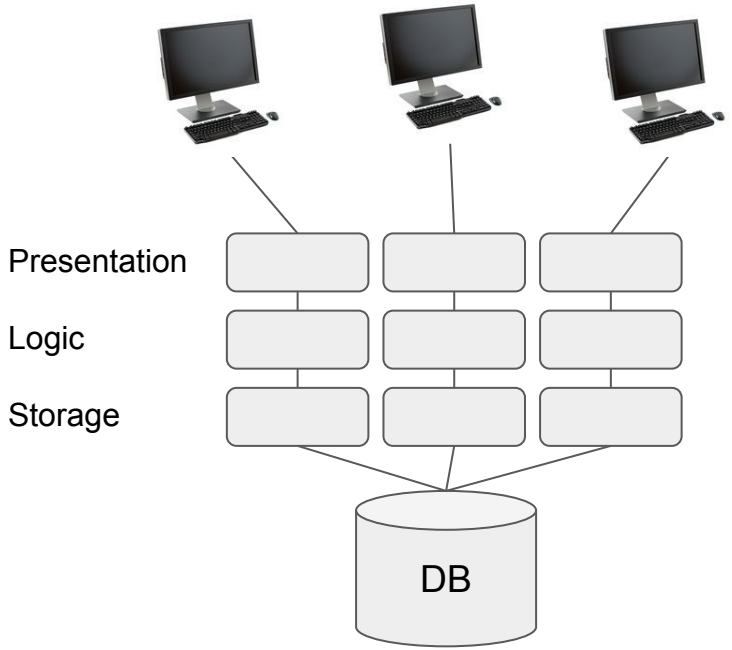
All businesses will be impacted

Relation is not optional

C.f. Kodak, Hasselblad

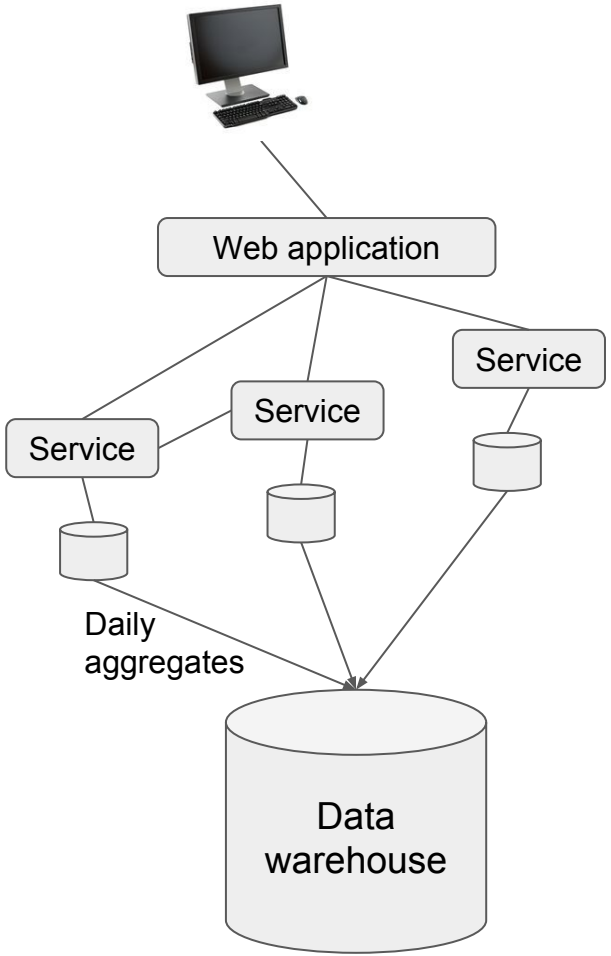
Data-centric systems - 1st generation

- The monolith
- All data in one place
- Analytics + online serving from single database
- Query current state

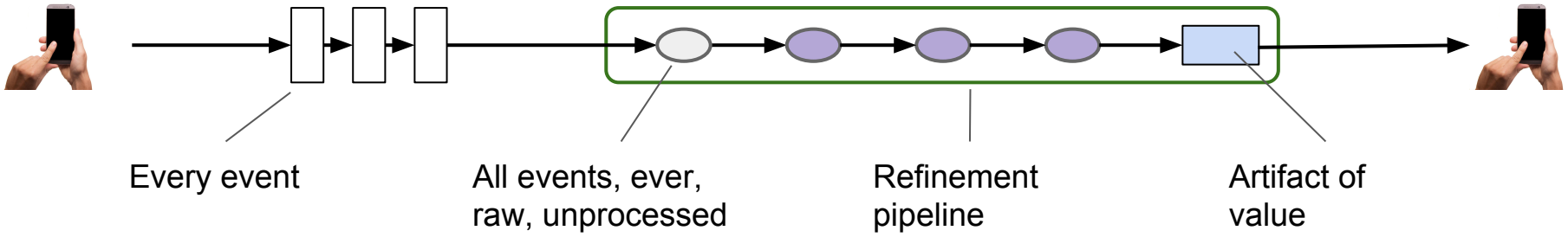


Data-centric systems - 2nd gen

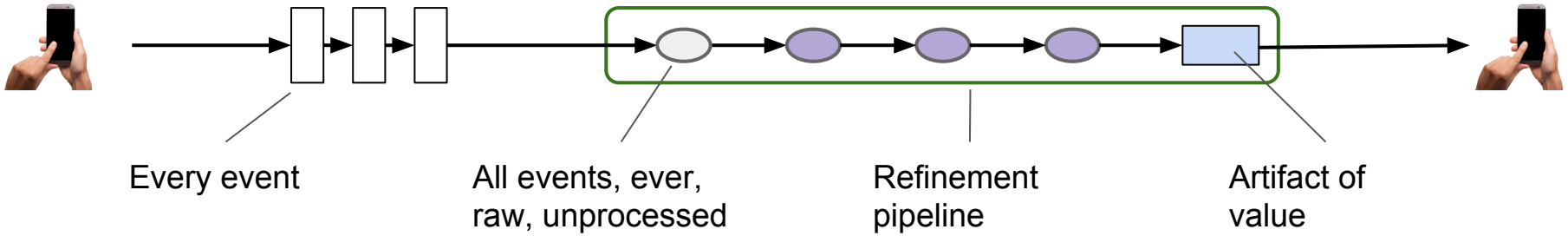
- Service oriented / microservices
- Collect aggregated data from multiple online systems to *data warehouse*
- Aggregate to OLAP cubes
- Analytics focused
- Query history of aggregations



Event oriented - 3rd generation



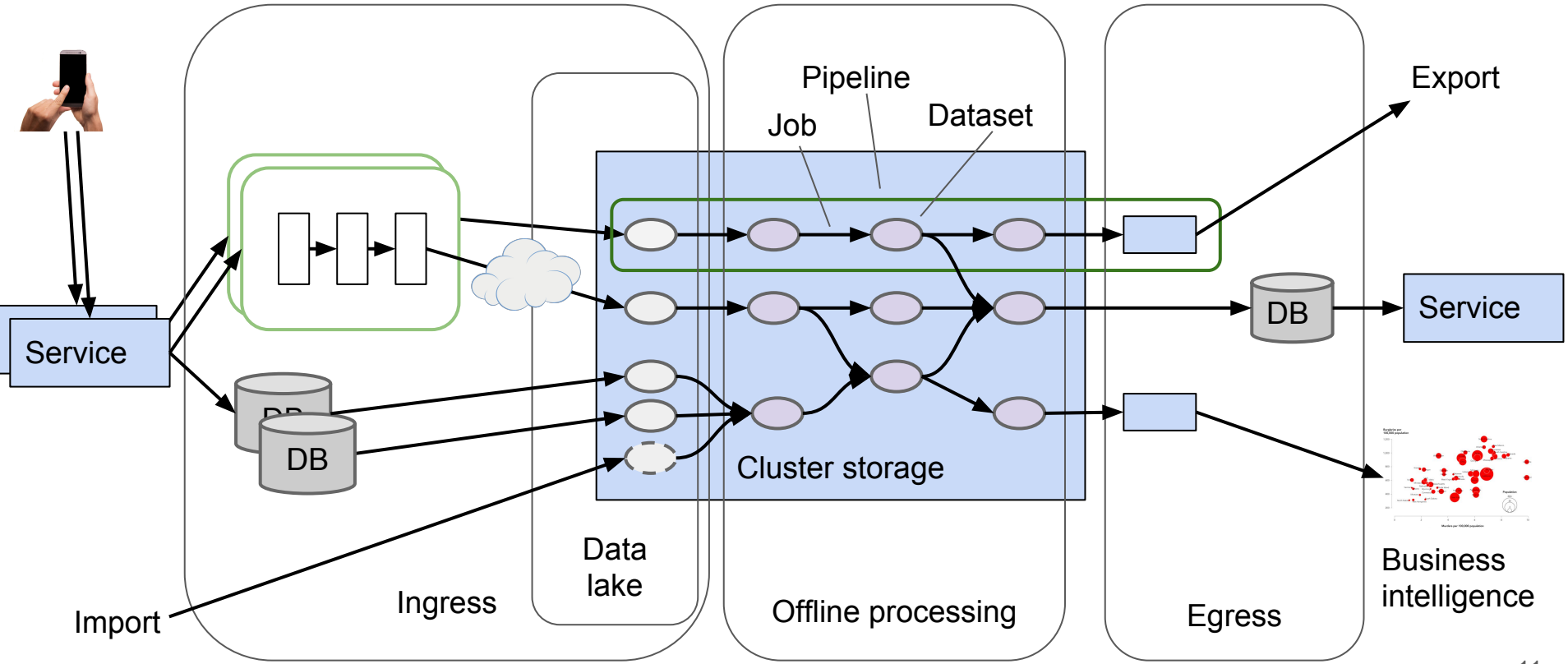
3rd generation - event oriented



- Motivated by
  - New types of data-driven (AI) features, served instantly
  - Quicker product iterations
    - Data-driven product feedback (A/B tests)
    - Fewer teams involved in changes
  - Robustness - scales to more complex business logic
  - Simple / homogenic - easier to comply with (privacy, financial) regulations
  - Detailed business insights
  - Scales to ~100M entities

Enables disruption

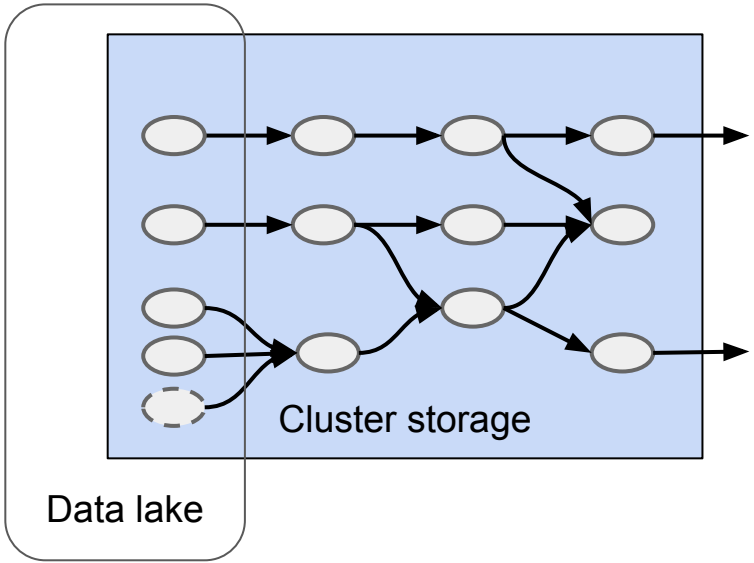
Data processing at scale



The data lake

Unified log of events + DB snapshots

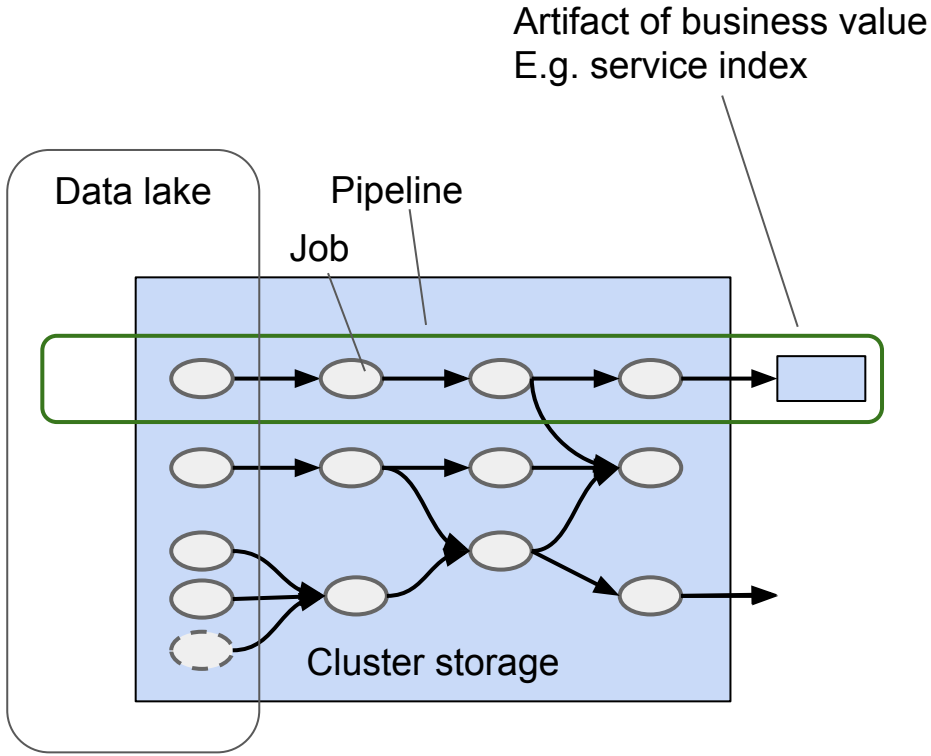
- Immutable datasets
- Raw, unprocessed
- Source of truth from batch processing perspective
- Kept as long as permitted
- Technically homogeneous



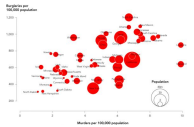
Batch processing

Gradual refinement

1. Wash
   - time shuffle, dedup, ...
2. Decorate
   - geo, demographic, ...
3. Domain model
   - similarity, clusters, ...
4. Application model
   - Recommendations, ...



Business architecture example, e-commerce



Events

User action

Merchant action

Partner event post

State (DBs + import)

Users

Inventory

Tag mgmt

OLAP cubes

Aggregate

Personalisation: collaborative filter

Hourly batches

Window, join, decorate

Click-through

Sessions

Behavioural pattern match

Inventory changes

My things events

Anomaly signals

Refined events

Monthly full

Daily delta

Item / item matrices

User / item matrices

Approx nearest neighbour

Item -> item suggestions

User -> item suggestions

Ranked push candidates

Inventory notification

Email reachout

Bundle suggest

Fraud candidates

Prepared for serving

User A: Item suggestion list
User A: Bundle suggestion list

Item B: Item suggestion list
Item B: Bundle suggestion list

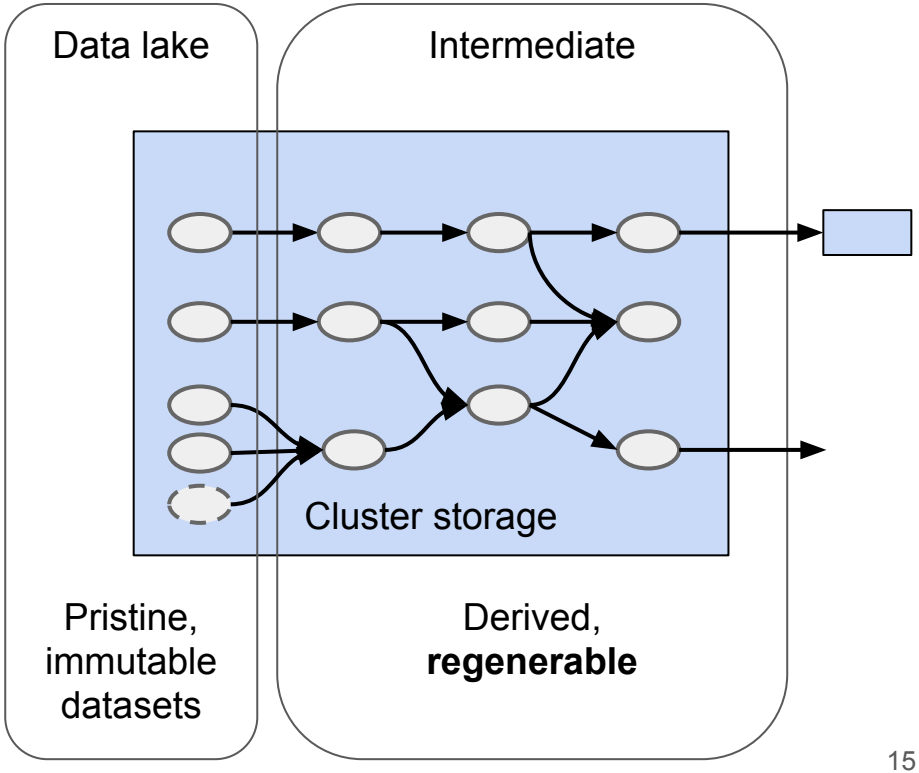
Merchant C: Fraud suspicions

Online multiquery, rank, select

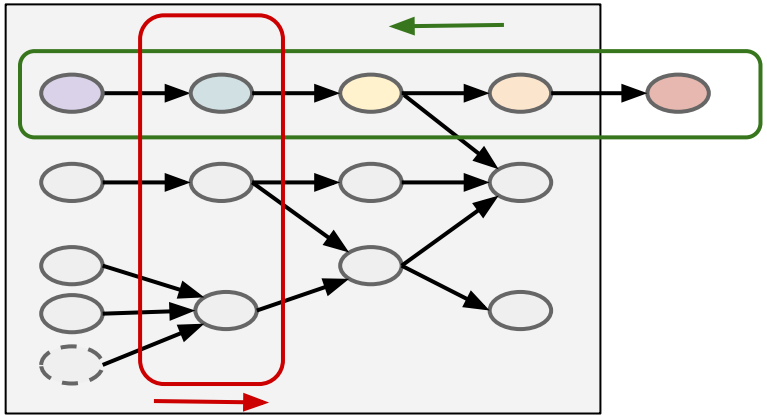
Pipelines

- Things will break
  - Input will be missing
  - Jobs will fail
  - Jobs will have bugs

- Datasets must be rebuilt
- Determinism, idempotency
- Backfill missing / failed
- Eventual correctness



Organisational dimensions



Waterfall, one team delivers to next

- Time to business value, low agility

Business value driven

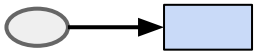
- Cross-functional teams with mission from raw data to value
+ Focus on value
+ Communication between functions
- Duplication
- Cross-cutting concerns
  - Technical productivity
  - Privacy & compliance
  - Security
- Staffing

Foundations of data teams

Product owner

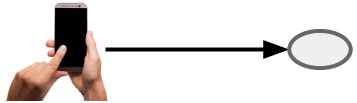
Business developer / project manager
+ data tech & science aware
Protector of ROI

Mutual
respect!



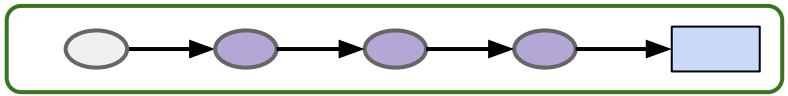
Data scientist

Analyst / statistician
+ capable programmer



Data engineer

Backend engineer
+ data wrangler



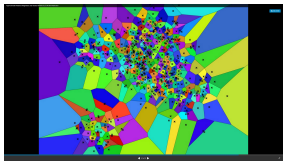
Use case - track recommendation

Entities

Tracks

2	4	1	1	5	2
0	1	7	1	0	6
5	2	9	0	3	0
3	8	0	6	0	7

Nearest neighbours



e1	e1782,e9823,..
e2	e3,e931,..
e3	e2,e8258,..
e4	e5243,e2802,..
...	...

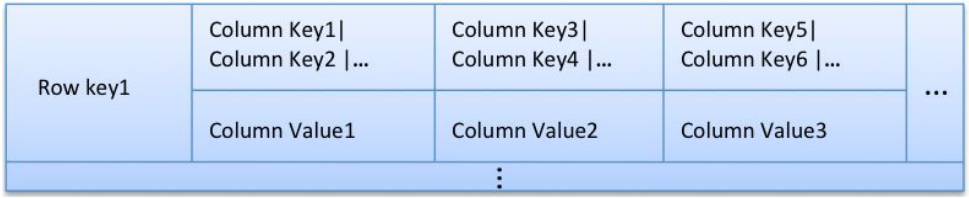
Distance = *cos(v1, v2)*

Entities = Tracks / Albums / Artists / Playlists / Users

~1000 neighbours

Use case - denormalise

e1	{0: e1782,”U2”, ...
e2	{0: e3,”Robyn”, ...
e3	{0: e2, “Röyksopp”, ...
e4	{0: e5243, “Cure”, ...
...	....



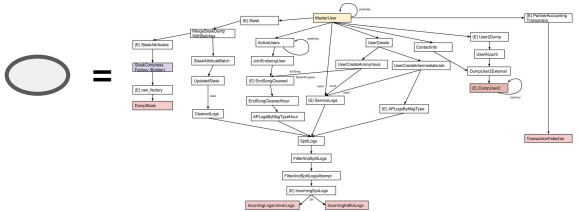
Credits: Ebay

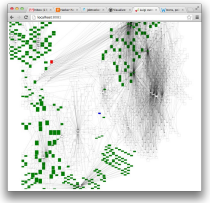
Requirements:

- Single dimension paging
- Read scalability
- Write scalability
- Reliability
- DC replication
- Availability over consistency

→ Cassandra

Example: Spotify





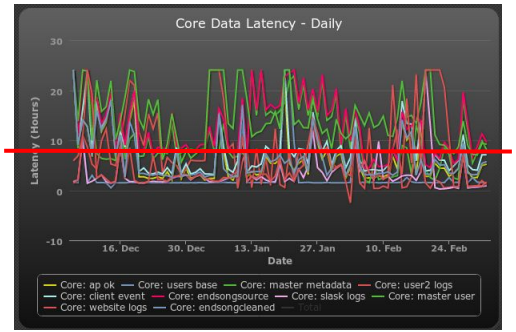
- 100M active users, 1B streams / day
- 60 TB/day ingress
- 2500 nodes Hadoop, 100PB, 100TB mem
- 20K jobs / day, 2K unique
- 600K BigQuery queries = 500 PB / month
- 500 people touch data daily

- Autonomous team and tech culture
- Mature, organic data platform

\+ Business-driven pipes, enabled teams

\- Productivity, end-to-end agility, privacy, stability, duplication, security

Morning coffee



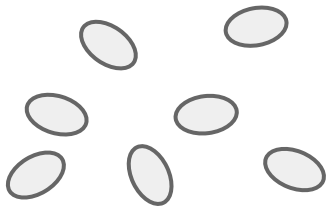
Anti-patterns

Enterprise edition

- Buy big Hadoop cluster
- Put all our data in a lake, wait for gold
- Army of consultants
- Assuming vendor solution is sufficient
- Disconnect business & data engineers
- We have lots of data, so we must have Big Data Things
- Excessive security & control

Startup edition

- Departments jealously guard their data
- Build for a scale not yet experienced
- Just hack things
- Engineering variety debt
- Drown in technical debt
- Assuming cloud solution is sufficient
- Teams choose different technical paths
- No end-to-end flows / agility



Solved

- Scaling out to many machines
- Performance / utilisation
- Hardware failures
- Software failure (bugs), batch processing
- Machine learning at scale
- Micro development process

Not solved

- Software failures, streaming
- Operations / human failures
- Scaling out to many developers
- Scaling out to complex pipelines
- Code evolution
- Macro development process
- Data quality process
- Privacy protection
- Data-driven product development
- Data discovery / lineage
- Human-in-the-loop