discodex: intuitive data indexing

Erlang User Conference, Stockholm, 2009

Jared Flatow
Ville Tuulos

state of disco

Disco 0.3 highlights

- Easier installation
- New fair scheduler
- Scales better to terabyte-scale datasets

Coming on the pipeline

- Embedded web server (mochiweb) for even easier installation
- Enhanced management of jobs and resulting data (tagging)
- Streaming results
- IO / network scheduling
- *adhoc data analysis & random data access*

big data

many huge (giga/terascale) datasets consist of lots of individual data records

- data is collected incrementally, and never deleted
- samples from an experiment or survey
- e.g. server logs, netflix training set, wikipedia, dna sequencing

some operations on big data are more expensive than others

- properties which have global dependencies are more expensive
- properties which are completely local to individual records are cheaper
- *we can usually precompute indices to speed up downstream operations*

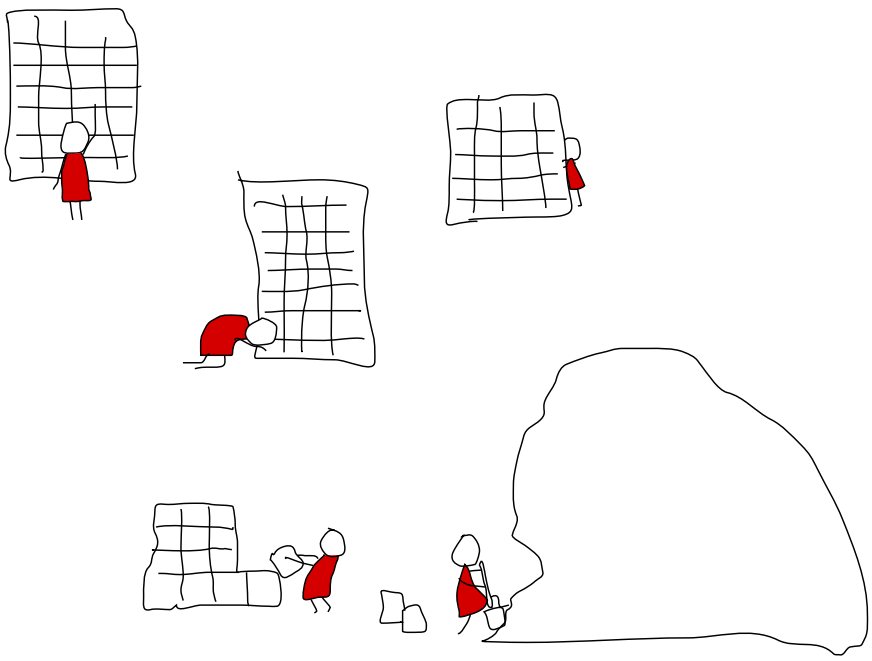
wishlist for big data infrastructure

- random access in arbitrary dimensions
- persistent distributed storage
- real-time + low-latency reads
- as-lazy-as-possible evaluation
- heterogeneous k/v scale (bytes to gigabytes)
- efficient multi-dimensional queries/joins ??
- pure and simple interface

the data storage landscape

- **mutable k-v stores**
  - e.g. dynamo/berkeleydb, tokyo cabinet, etc.
  - only support single-key lookup
- **bigtable-like (column-based, semi-structured, distributed hash table)**
  - e.g. hypertable, hbase, cloudstore, hstore, etc.
  - highly complex, difficult to get right
  - no mature (open-source) implementations
- **relational databases**
  - lots of overhead, both maintenance and transactional
- **erlang-specific**
  - e.g. mnesia, dets
  - not built for scale/high-performance
  - no external interface
- **document-based stores**
  - e.g. couchdb
  - not meant for huge data

x: intuitive data indexing



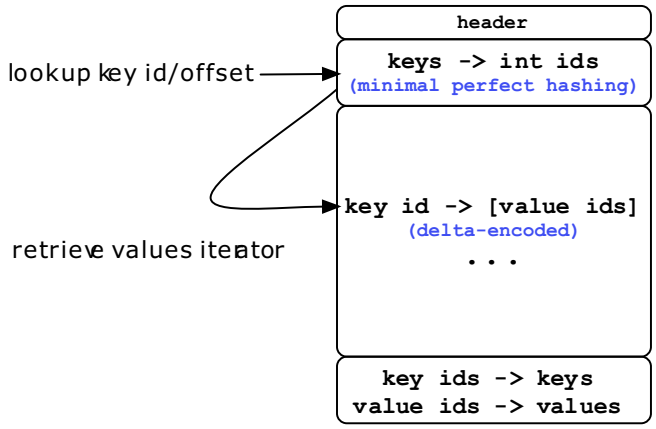
© 2009 Nokia Research Center

discodb

- low-level C data structure
- maps key -> multiset(values)
- immutable + persistent: write once to a file
- Python/erlang wrappers: api = dictionary + cnf

discodb format

designed for lightning fast random-acces



discodb.erl

ets-like Erlang binding to discodb

- `ddb:new(), ddb:add(Ddb, Key, Val)`
- `ddb:lookup(Ddb, Key), ddb:query(Ddb, CnfQuery)`
- `ddb:select(Ddb, MatchSpec)`

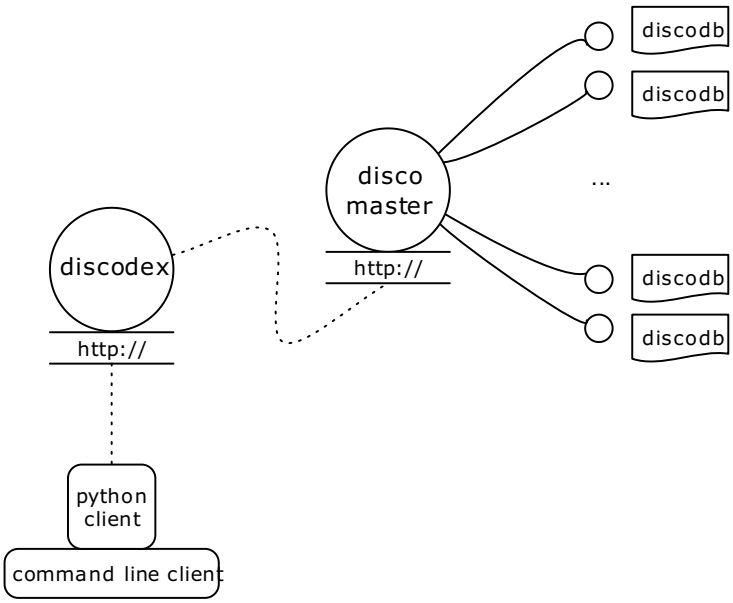
Lazy query evaluation with continuations

x: intuitive data indexing

discodex

- distributed discodbs form indices for data
- disco jobs create indices/harvest query results
- build/query indices through RESTful API
- dead simple command line/Python interfaces
- indexing parameterized by parser/demuxer/balancer functions
- supports querying billions of keys in real-time

discodex design




```
> cat dataset | discodex index
> discodex get <index> | discodex query <query>
```


live demo!

questions?