

Disco: Getting Down with Big Data

SciPy 2010: Austin, TX

Jared Flatow

© 2010 [Nokia Research Center](#)

Data Insight @ NRC Palo Alto

Taneli Mielikäinen
Ville Tuulos
Jared Flatow
Prashanth Mundkur
Deepak Jagdish

© 2010 [Nokia Research Center](#)

big data

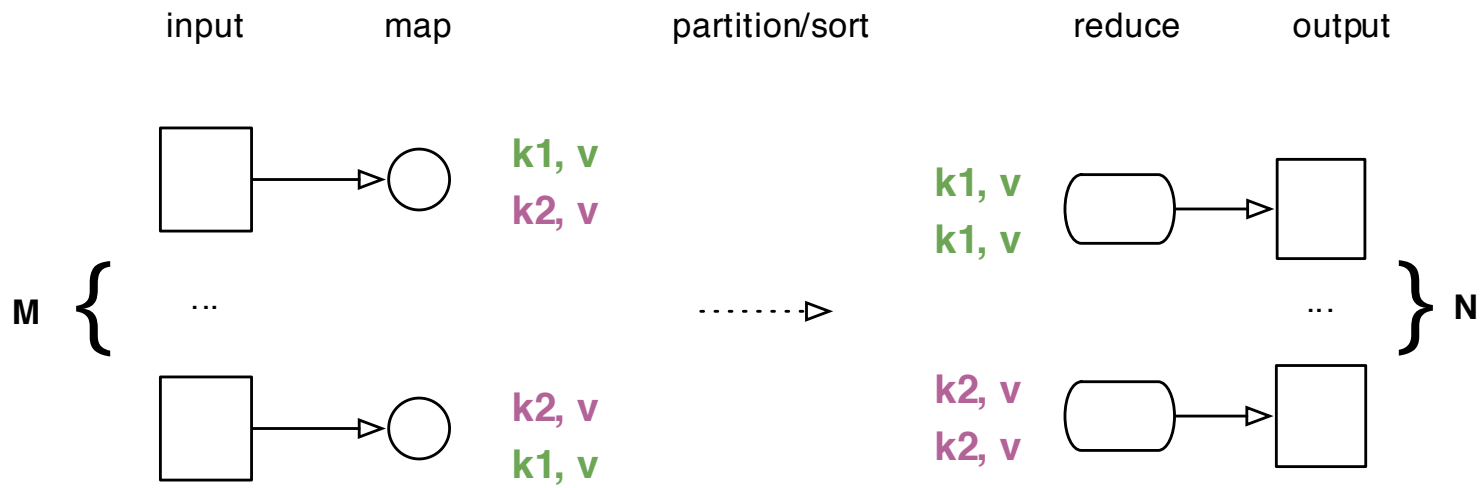
many huge (giga/terascale) datasets consist of lots of homogenous data records

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

many analyses operate on subsets of data for which some property holds

- operations can be sped up by precomputing indices for properties of interest
- e.g. requests from N900, movies rated by customer, articles modified on date, reads containing substring

mapreduce



disco

- erlang for orchestrating control
- python for operating on data
- *batteries included*:
 - utilize favorite python libraries
 - includes additional goodies to make working with distributed data easier

an example

```
# key    0.6
# key2   0.4
# key    0.4

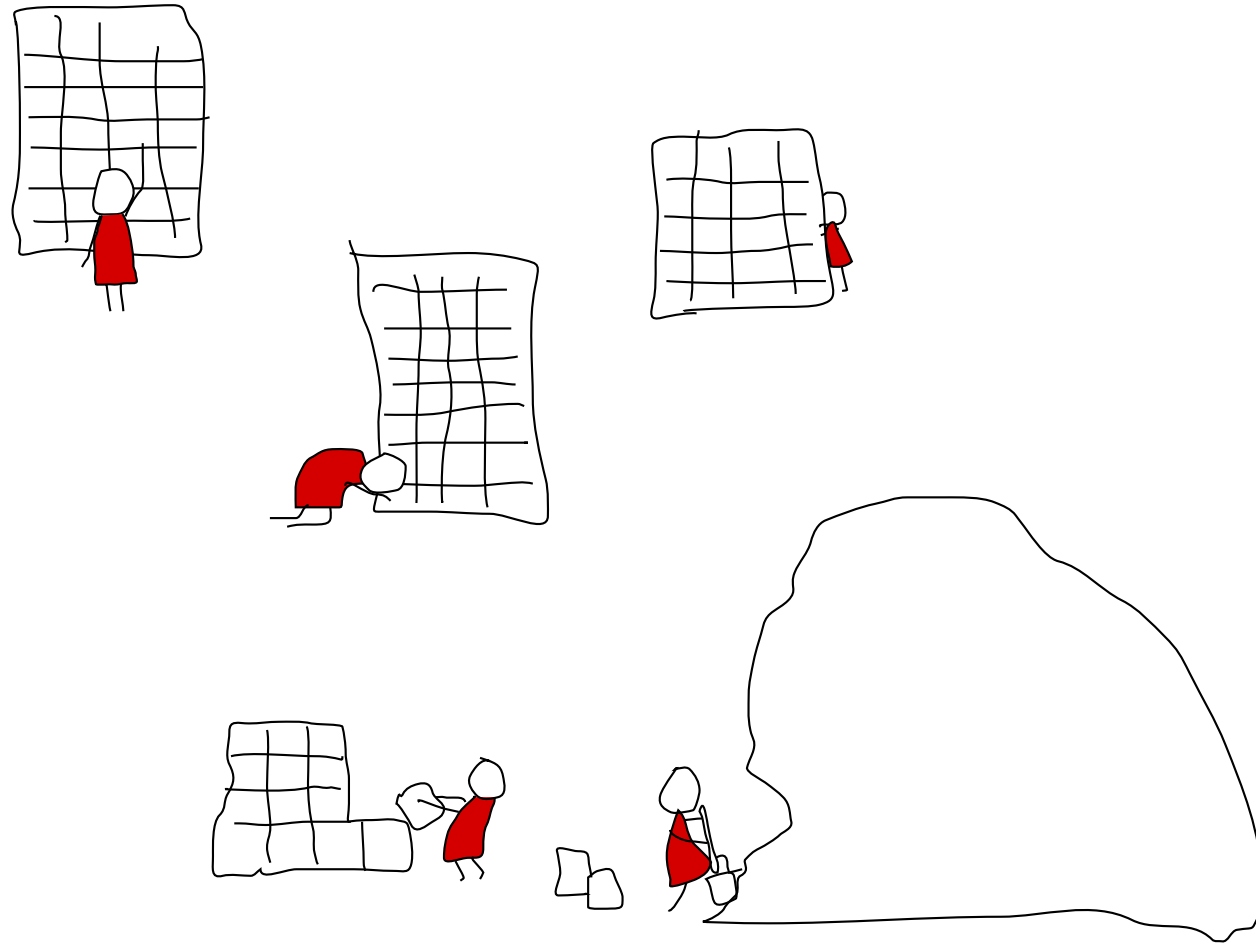
def map(entry, params):
    fields = entry.split('\t')
    yield fields[0], float(fields[1])

def reduce(iterator, out, params):
    from disco.util import kvgroup
    from numpy import var
    for k, vs in kvgroup(iterator):
        out.add(k, var(list(vs)))

# key    0.01
# key2   0
```

disco distributed "filesystem" (ddfs)

- distribute data:
 - increase storage capacity by adding nodes
 - processing can be scheduled on nodes without transferring data
- replicate data:
 - fault-tolerance
 - more options for scheduler



discodb

- lightning-fast key -> values mapping
- low-level C data structure
- Python wrapper: api = dict + cnf
- immutable + persistent
 - write once to a file
 - read using memory-map

© 2010 [Nokia Research Center](#)

discodex

- disco + ddfs + discodb
- simple ReSTful interface
- random access in real-time
- as-lazy-as-possible evaluation
- heterogeneous k/v scale (bytes to gigabytes)

© 2010 [Nokia Research Center](#)

indexing the oil spill data

```

def parser(fd, size, url, params):
    import csv, re
    from discodex.mapreduce import Record
    def format_date(date, date_re=r'(?P<month>\d+)/(?P<day>\d+)/(?P<year>\d+)'):
        match = re.match(date_re, date)
        return '%04d-%02d-%02d' % (int(match.group('year')),
                                   int(match.group('month')),
                                   int(match.group('day')))
    for n, row in enumerate(csv.reader(fd)):
        if any(row):
            if n == 0:
                source = row[0].strip().replace(',', '')
                for type in ('Air', 'Oil', 'Sediment', 'Water'):
                    if type in source:
                        break
            elif n == 1:
                fieldnames = [str(name.lower()) for name in row if name]
            elif n > 1:
                fields = [str(field.decode('utf-8', 'ignore')) for field in row]
                record = Record(n=n,
                                source=source,
                                type=type,
                                **dict(zip(fieldnames, fields)))
                if hasattr(record, 'date'):
                    record.date = format_date(record.date)
            yield record

```

live demo!

type:Sediment  [load data](#)

© 2010 [Nokia Research Center](#)

questions?

more information at www.discoproject.org

© 2010 [Nokia Research Center](#)