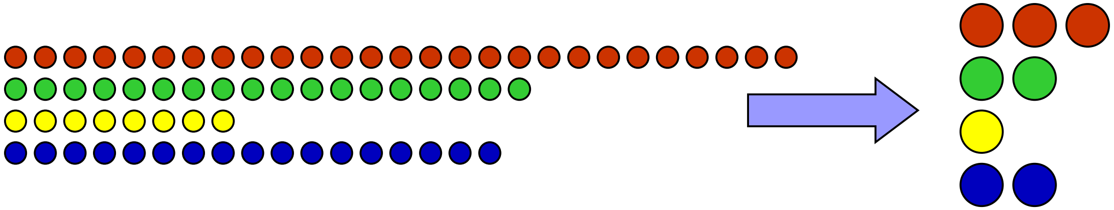
Small Summaries for Big Data



Ke Yi

HKUST

yike@ust.hk

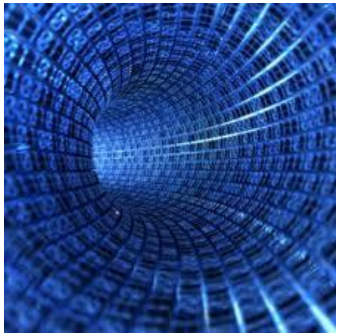
MASSIVE Data

- Massive data is being collected everywhere: business, technology, scientific research, etc.
- Examples:
  - TIGER/Line data set: 16.7 million road segments
  - LIDAR data set: 500 million points for just Neuse River Basin (14GB)
  - AT&T phone call database: 20TB
  - Google indexes 20 billion web documents
- Keep growing!

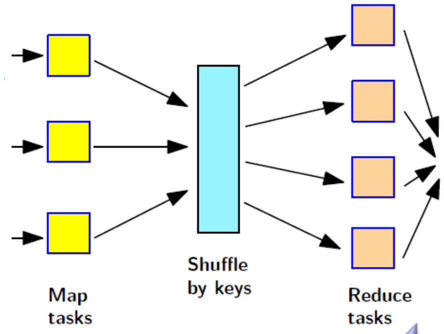
Big
~~Massive~~ Data Algorithms

External Memory Algorithms

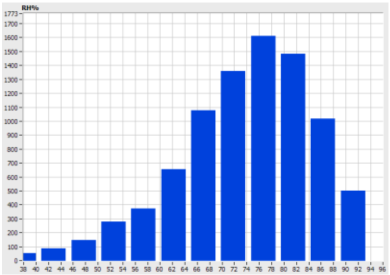
Data Stream Algorithms



Parallel/Distributed Algorithms



Sublinear Algorithms



1988

1999

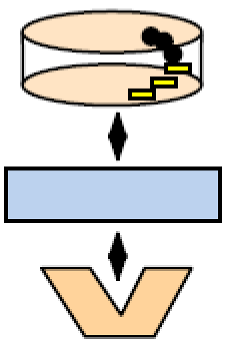
2006

2015

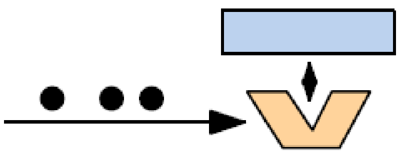
Theory → Practice

Example: Random sampling

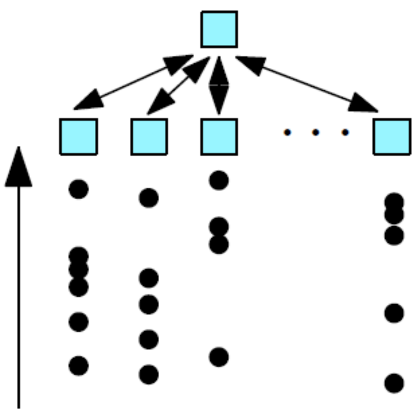
- Random sampling in standard model is trivial.
- Becomes challenging when…



External memory



Streaming



Distributed Streams

Query sampling

- Sample from the query results without evaluating the query!

Sampling from a range query (SIGMOD'15 best demo award)

Sampling for joins

```
select
        n_name,
        sum(l_extendedprice * (1 - l_discount)) as revenue
from
        customer,
        orders,
        lineitem,
        supplier,
        nation,
        region
where
        c_custkey = o_custkey
        and l_orderkey = o_orderkey
        and l_suppkey = s_suppkey
        and c_nationkey = s_nationkey
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = '[REGION]'
        and o_orderdate >= date '[DATE]'
        and o_orderdate < date '[DATE]' + interval '1' year
group by
        n_name
order by
        revenue desc;
```

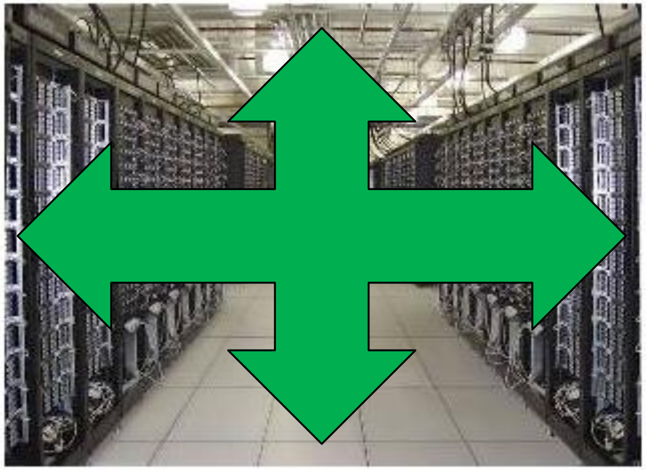
- TPC-H Benchmark Query
- 6 tables
- 10GB data
- Takes >1 hour in Oracle
- Our algorithm finishes in <10 seconds[1]
- Now working on integration into PosgreSQL

[1]Returns an answer with $\pm$1% error

How NOT to do it

- Suppose there are 2 tables:
  - Companies (CompanyID, Nation)
  - Orders (OrderID, SellerD1, BuyerID2, Revenue)
- Say, the query asks for the total revenue of all orders made between a company in China and another in the US
- Simple random sampling:
  - Take a 0.01% sample (1MB data) from Companies
  - Take a 0.01% sample (1MB data) from Orders
  - Only get 1MB * 0.01% * 0.01% = 0.01 byte of joined data!

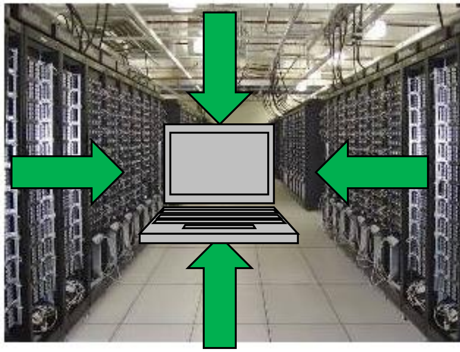
Scaling up / out computation



- Many great technical ideas:
  - Use many cheap commodity devices
  - Accept and tolerate failure
  - Move code to data
  - MapReduce: BSP for programmers
  - Break problem into many small pieces
  - Decide which constraints to drop: noSQL, ACID, CAP
- Scaling up comes with its disadvantages:
  - Expensive (hardware, equipment, **energy**), still not always fast

Scaling down data

- A second approach to dealing big data: **scale down the data!**
  - A compact representation of a large data set
  - Too much redundancy in big data anyway
  - What we finally want is small: human readable analysis / decisions
  - Necessarily gives up some accuracy: approximate answers
  - Often randomized (small constant probability of error)
  - Examples: samples, sketches, histograms, wavelet transforms
- Complementary to the first approach: not a case of either-or



Flavors of my research

- Focus on fundamental problems
  - Random sampling, hashing, sorting
  - New challenges arise in the “big data” era even for these fundamental problems
  - Don’t work on made-up problems just for writing papers
  - Don’t work on “n choose k” problems like “k nearest neighbor search in high dimensions using L1 metric on uncertain data with multiple query points and keywords using MapReduce”
- Only work on well-defined problems
  - Actually, the main challenge in many areas of CS (data mining, machine learning, etc) is to find the right definition.
- Don’t work on NP-hard problems