Hive
SparkSQL
SQL database ?

Mohit
Sarthak

MapReduce / Hadoop

- abstractions
  - fault tolerance
  - parallelism ("implicit")
  - scalability - scaling up (# processors)
- Why? - schemaless out (# of machines)

100s - 1000s

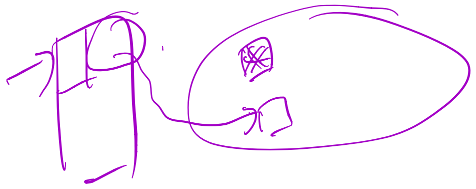
- heterogeneous

one cause stragglers
bad block
remapping

SKU
stock keeping units

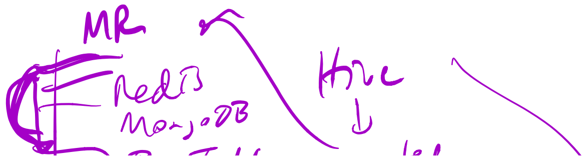
RAM
CPU MHz GHz
disk I/O

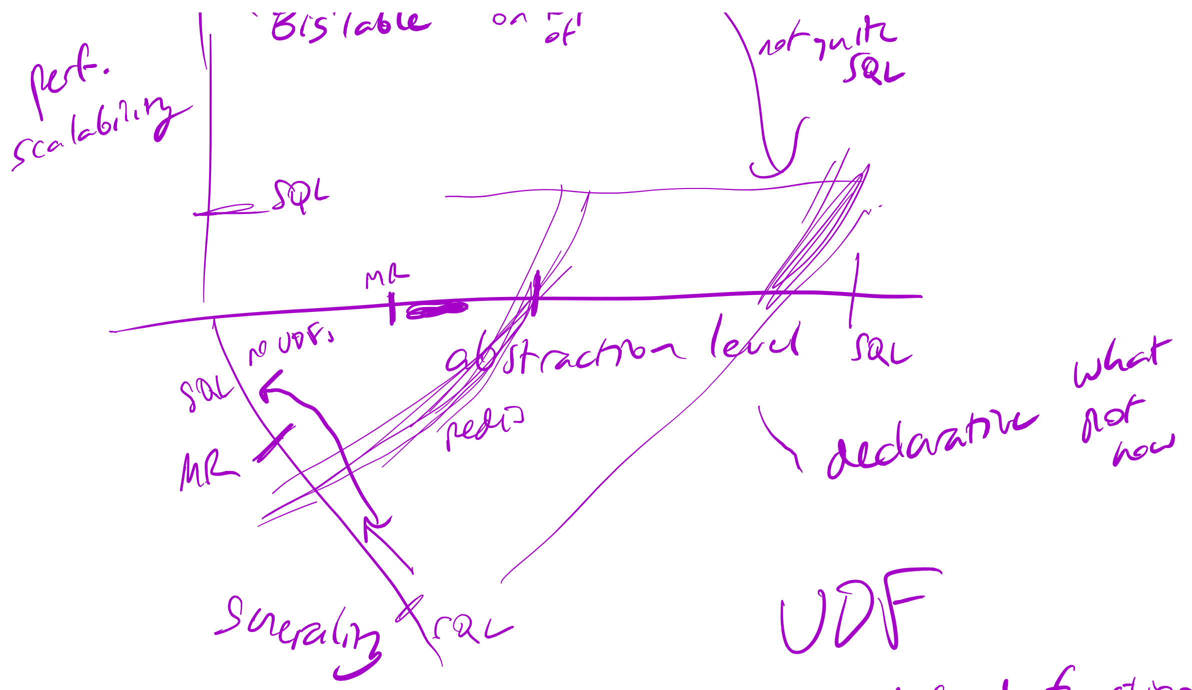
"slow death"



key-value stores
Redis data structures
JSON stores MongoDB

MR





UDF
user defined function

UDAF
aggregator

MR - low sen.
narrow API
map, reduce
+ iterate

HiveQL — subset

— no support for updating/deleting rows
in existing tables

+ UDF, UDAFs in Java

integration — JNI faster than Hadoop Java
coupling

JVM
.NET

—( )

+ streaming rows

```
ctx = new HiveContext();
users = ctx.table("users");
young = users.where(users("age") ≤ 21);
println(young.count());
```

```
SELECT count(users.age)
FROM users.
WHERE users.age ≤ 21;
```

— data mapping

tables $\longrightarrow$ map reduce files

HDFS

Hadoop

directories in HDFS

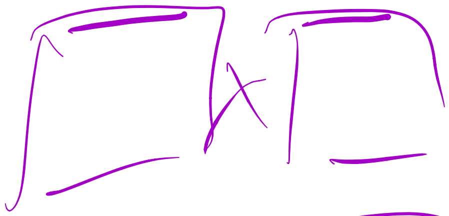
partitioning [/wh/T/ds=20090101/ctry=US

table

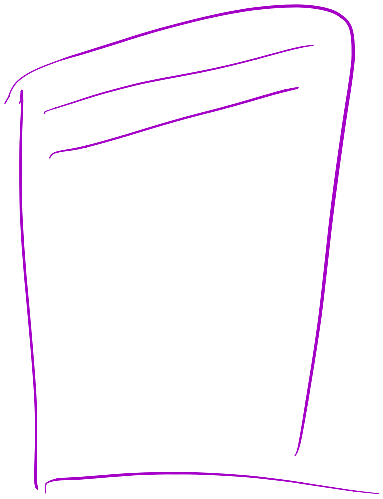
locality / enables distribution

Compile queries ~ FlumeJava
HiveQL

SELECT *
FROM —, —
WHERE (. . . .) predicate pushdown



— X —



SELECT ~~*~~ emp.name
FROM emp, contractors
WHERE emp.salary < 10000 emp
AND
contractors.salary < 10000 cont.

column pruning

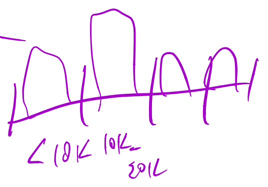
disadvantages

① disk I/O

→ Hadoop slow

② not full SQL

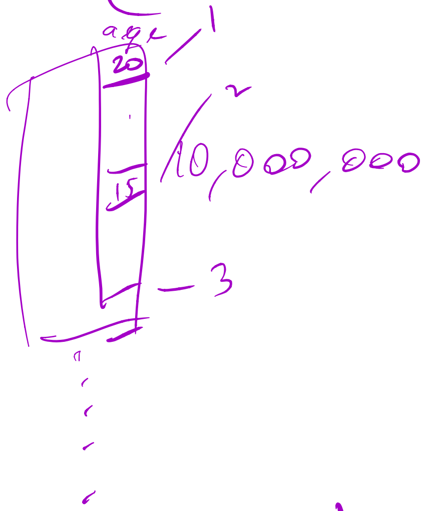
SAMPLING



buckets pruning

~~SELECT~~

(count young ppl)



N

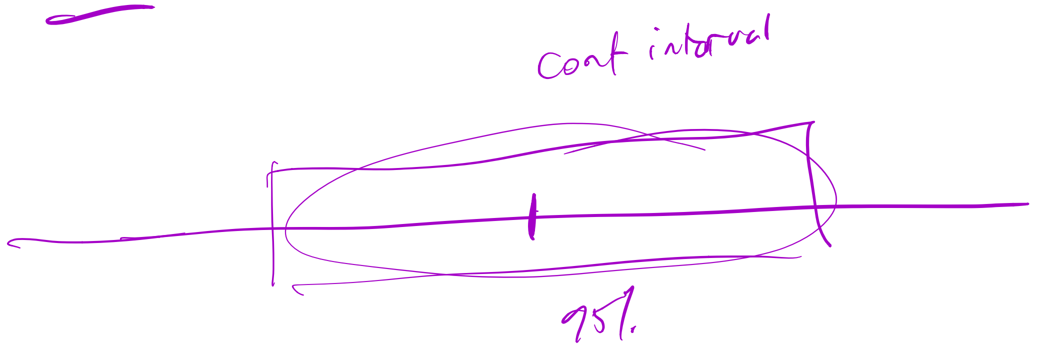
sample N

random downsample by 10,000

count 1,000 records

× 10,000

—



95000

~~0~~ ± 1000

"needle in a haystack"

X sampling

— UDFs / UDAFs

Java

which columns?
what predicate?

WHERE (f(x) ...)

BLACK BOX

SPARK SQL