Dremel: Interactive Analysis of Web-Scale Datasets

Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer,
Shiva Shivakumar, Matt Tolton, Theo Vassilakis
Google, Inc.

Presenter: Rui Wang

Background

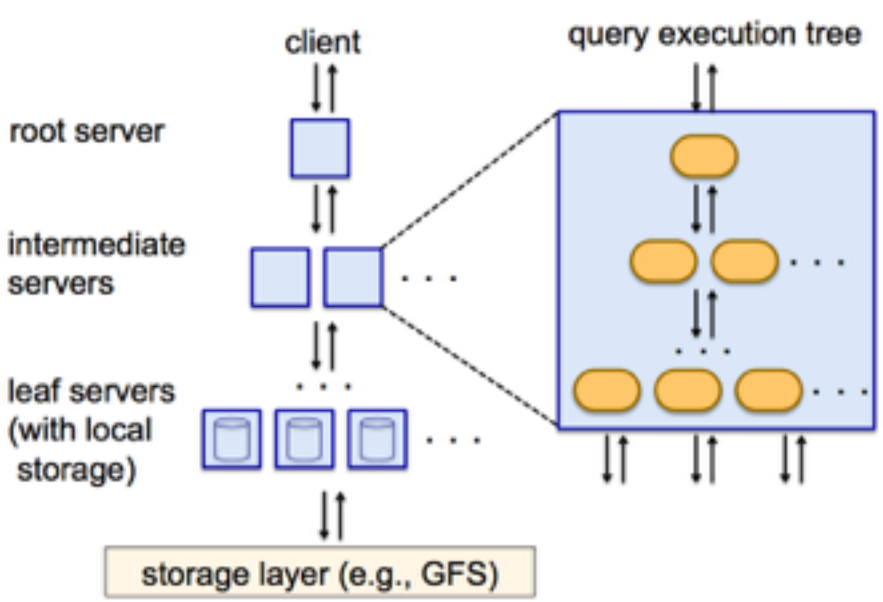
Analytic Service

Parellel Processing

Distributed File System

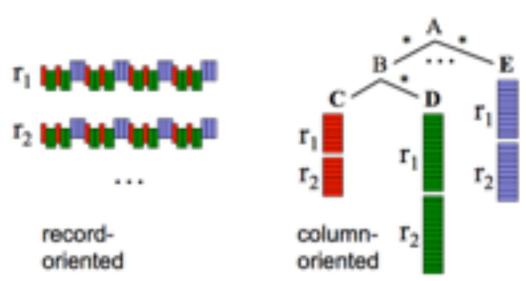
Is MapReduce enough for Large Scale Data Analysis?

Dremel



- Nested Column Store
- Serving Tree Engine
- Diverse Measures

Data Model



DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

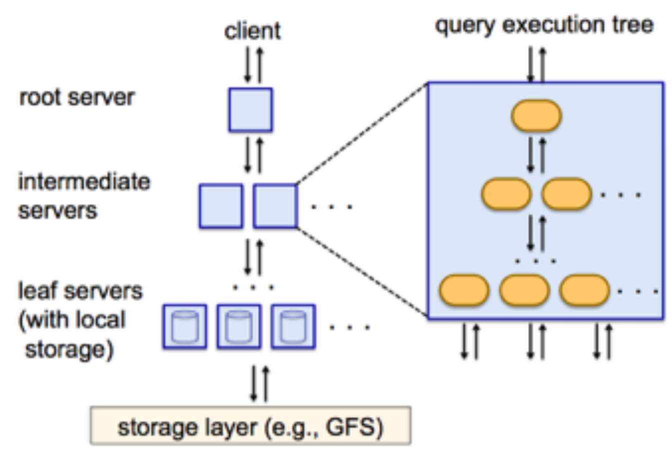
Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

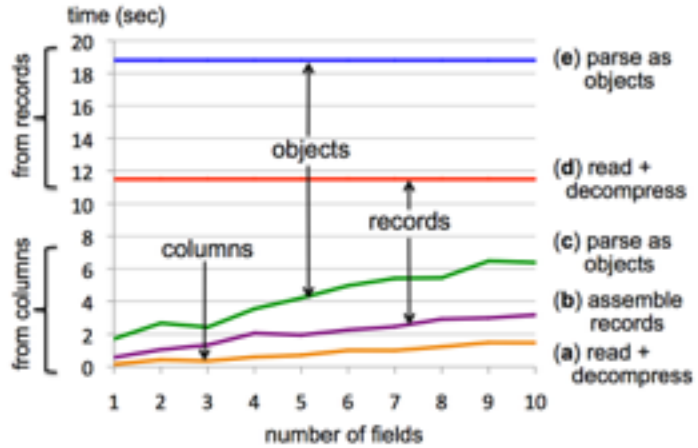
Query

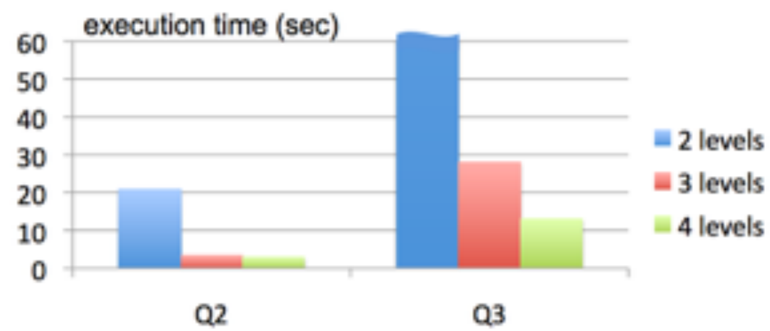


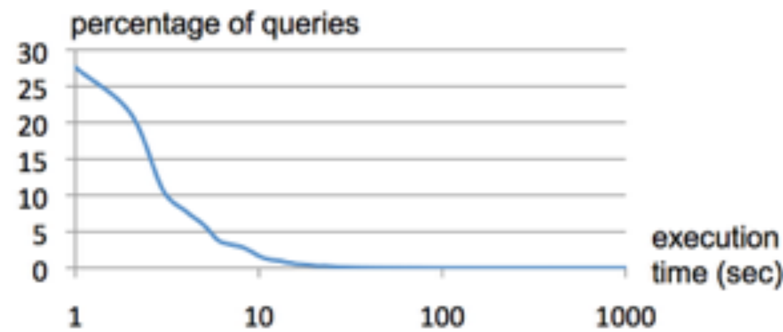
SELECT A, COUNT(B) FROM T GROUP BY A

SELECT A, SUM(c) FROM (R_1 UNION ALL ... R_n^1) GROUP BY A

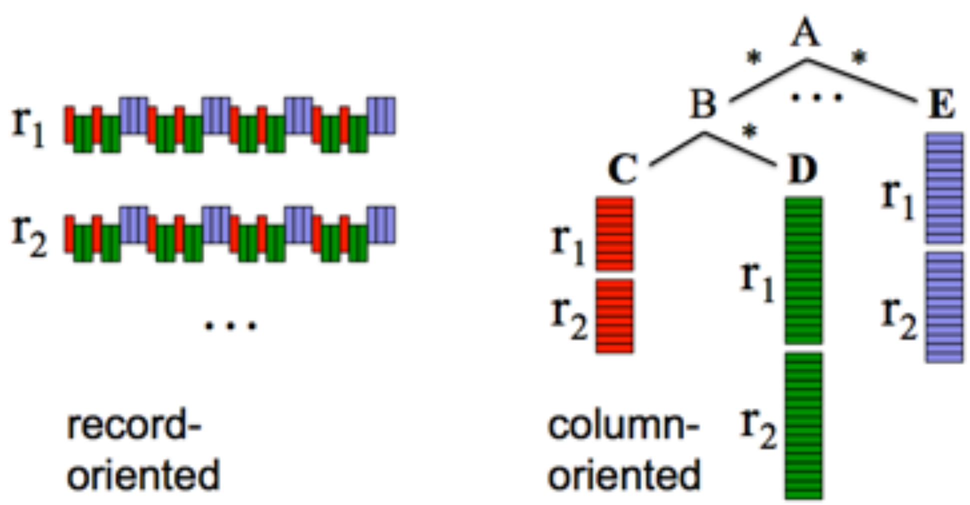
Experiments







Nested Columnar Storage



- Structured Format
- Efficient Encoding
- Fast Transform

Nested Columnar Storage

Repetition & Definition Level

```
DocId: 10                    r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
DocId: 20                    r2
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```

DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

Nested Columnar Storage

Repetition & Definition Level

```
DocId: 10                 r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
DocId: 20                 r2
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```

1 Name	2 Language	3 Code	Repetition
1	1	en-us	0
1	2	en	2
2	0	NULL	NULL
3	1	en-gb	1
0	0	0	NULL

Nested Columnar Storage

Repetition & Definition Level

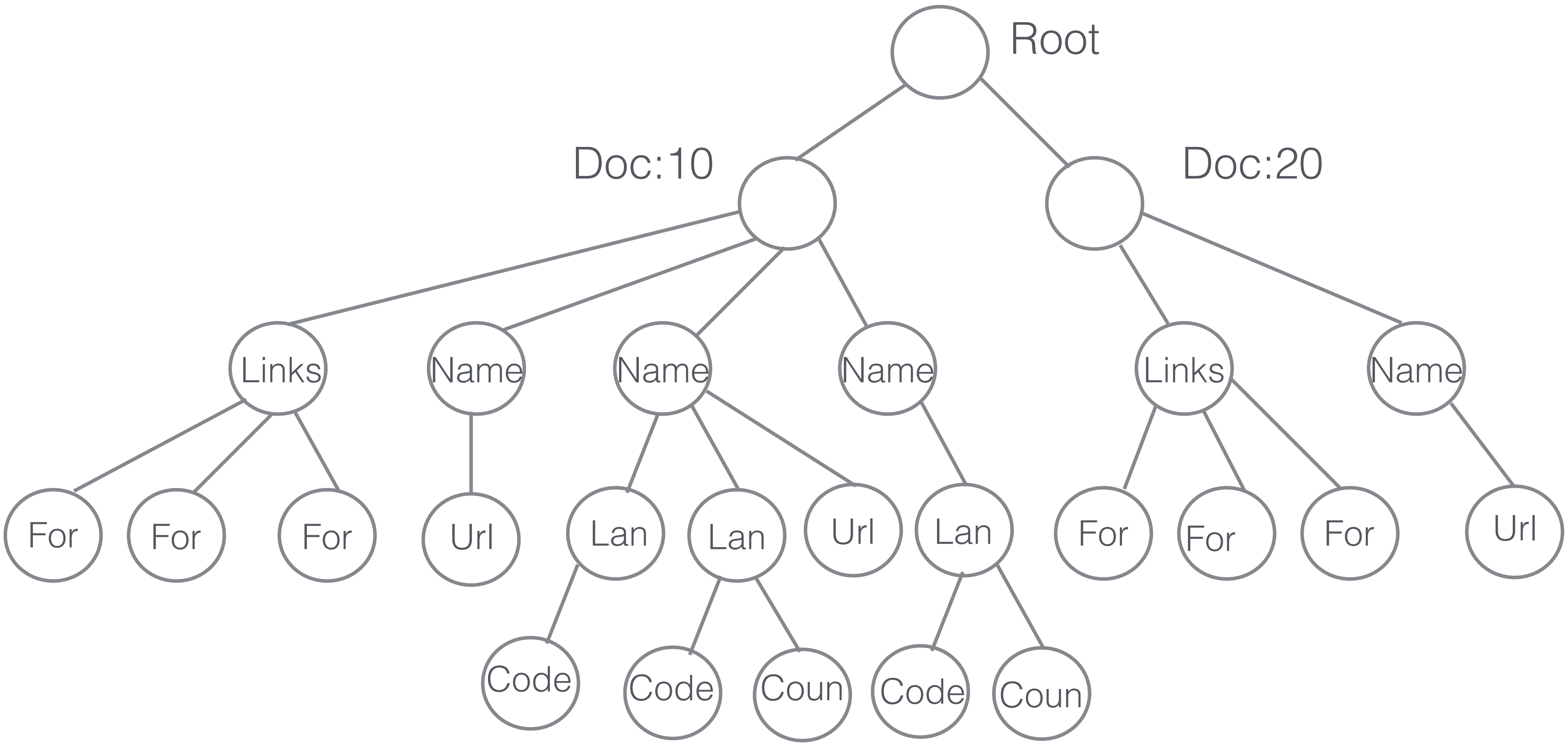
```
DocId: 10                r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
DocId: 20                r2
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```

1	2	Code	3	
Name	Language		Country	Definition
1	1		us	3
1	2		NULL	2
2	1		NULL	1
3	1		gb	3
1	1		NULL	1

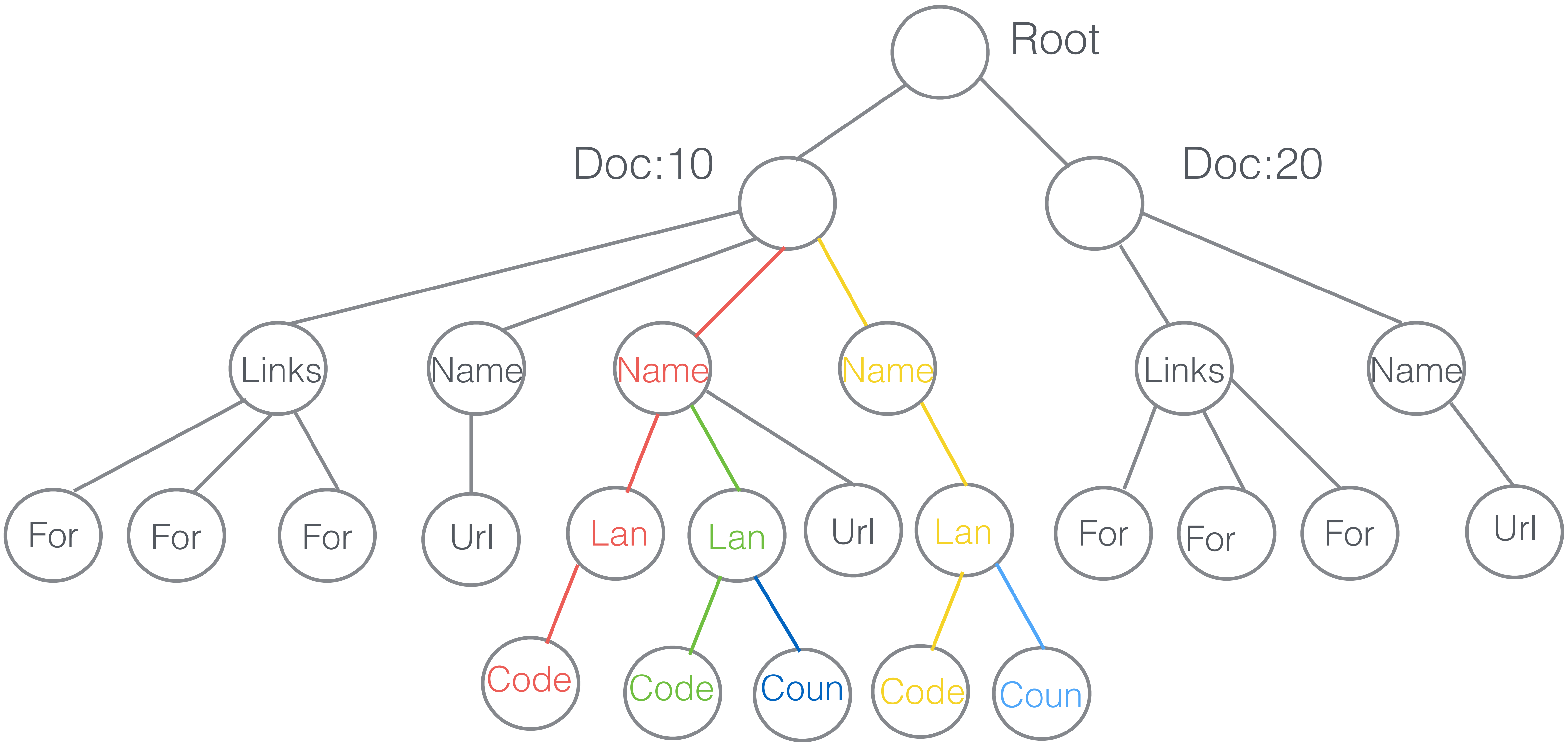
Nested Columnar Storage

Repetition & Definition Level



Nested Columnar Storage

Repetition & Definition Level

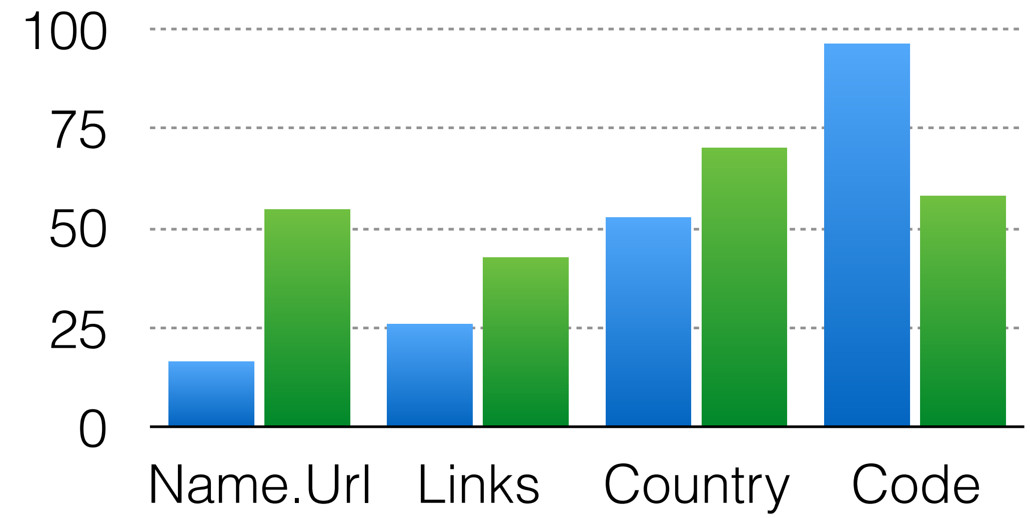


Repetition:	0	2	1
Definition:	2	2	2

Nested Columnar Storage

Split Records to Columns

```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
```

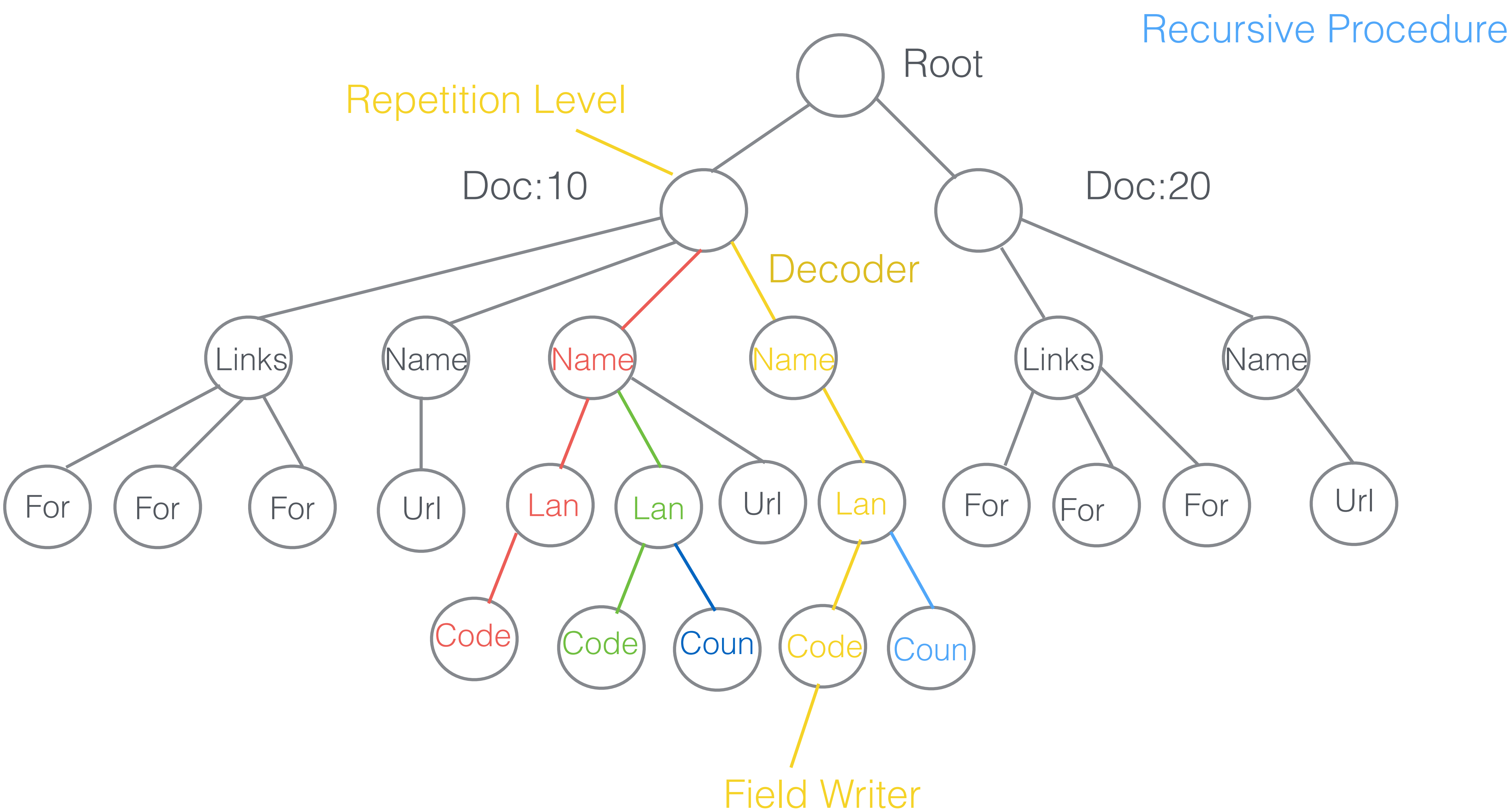


Protocol Buffer

How to Serialize to Columns?

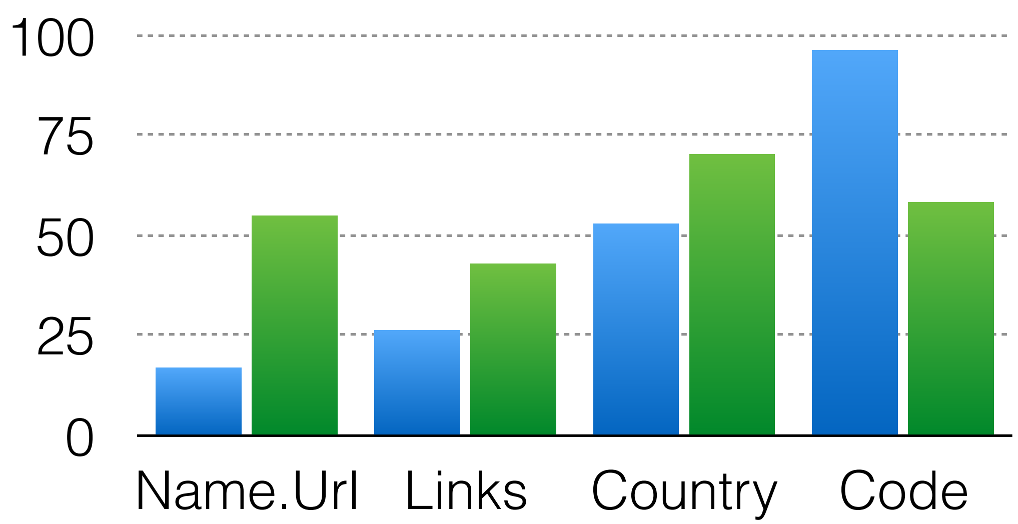
Nested Columnar Storage

Splitting Records into Columns



Nested Columnar Storage

Record Assembly



Columnar Storage

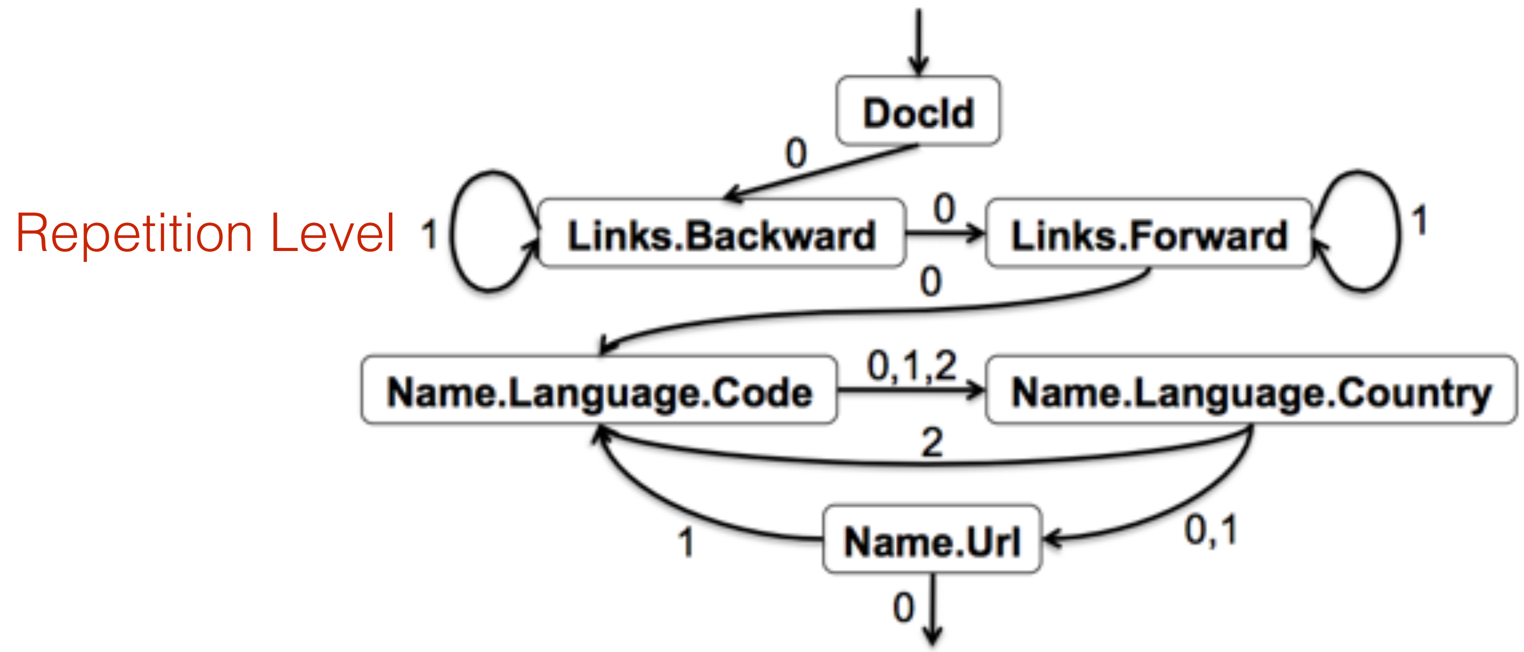
```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
```

Protocol Buffer

Nested Columnar Storage

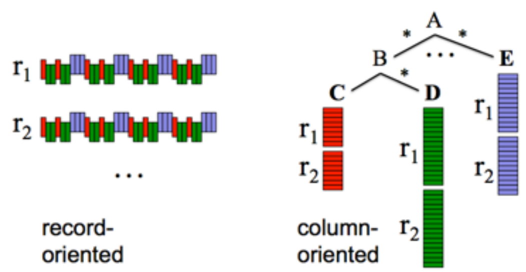
Record Assembly

```
DocId: 10                r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```



Finite State Machine

Data Model



DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

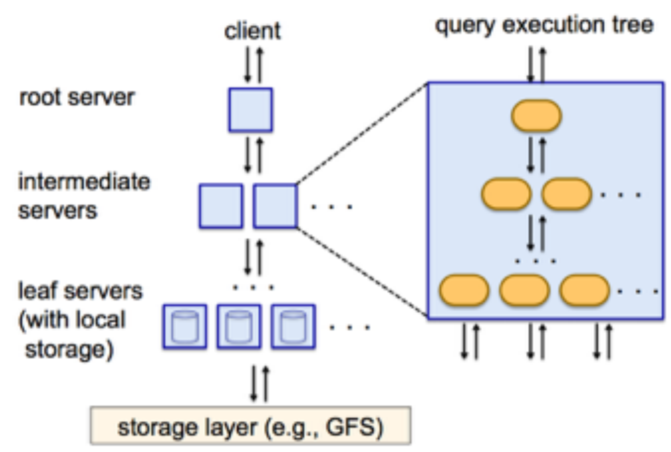
Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

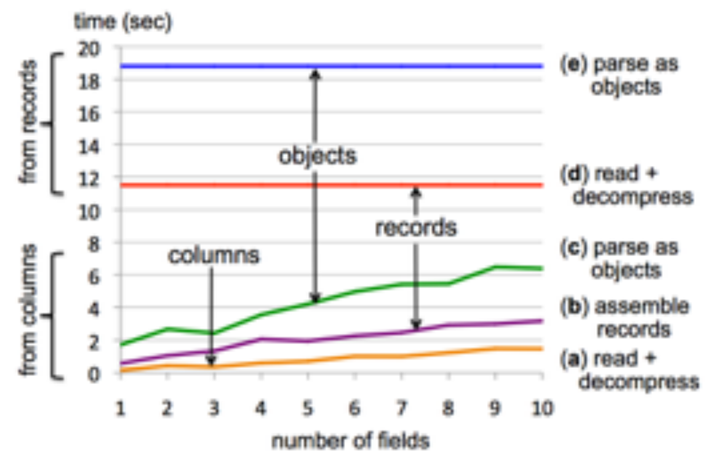
Query

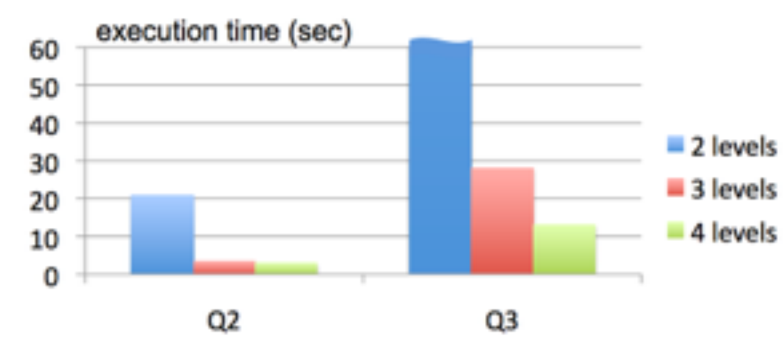


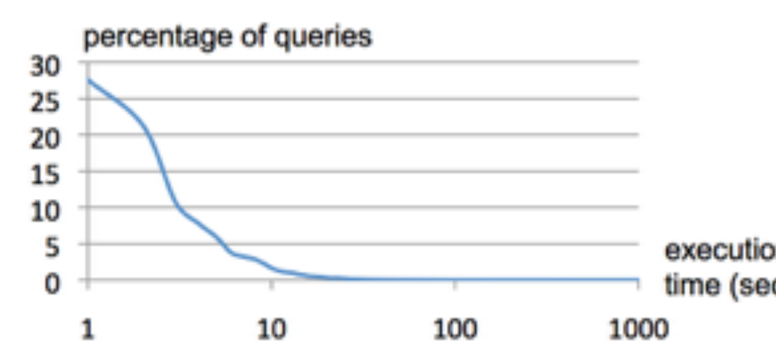
SELECT A, COUNT(B) FROM T GROUP BY A

SELECT A, SUM(c) FROM (R_1 UNION ALL ... R_n^{1}) GROUP BY A

Experiments







Query Language

```
SELECT DocId AS Id,
  COUNT(Name.Language.Code) WITHIN Name AS Cnt,
  Name.Url + ',' + Name.Language.Code AS Str
FROM t
WHERE REGEXP(Name.Url, '^http') AND DocId < 20;
```

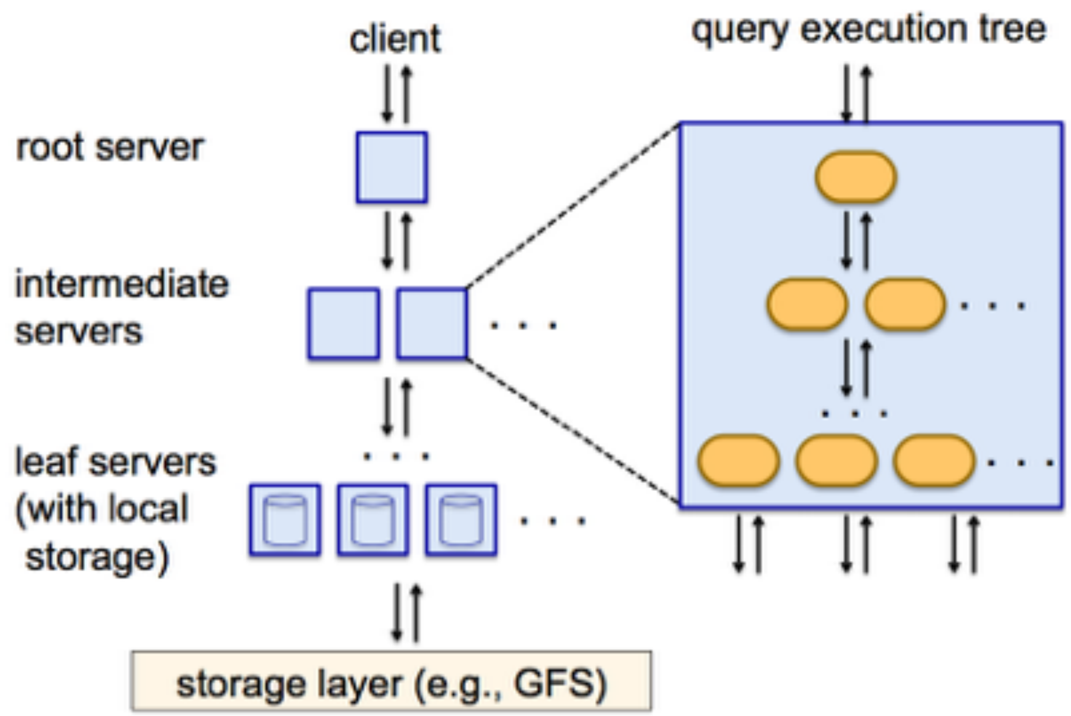
t_1

```
Id: 10
Name
  Cnt: 2
  Language
    Str: 'http://A,en-us'
    Str: 'http://A,en'
Name
  Cnt: 0
```

```
message QueryResult {
  required int64 Id;
  repeated group Name {
    optional uint64 Cnt;
    repeated group Language {
      optional string Str; }}}
```

Query Execution

Serving Tree Topology



A root server receives incoming queries

Reads metadata from the tables,

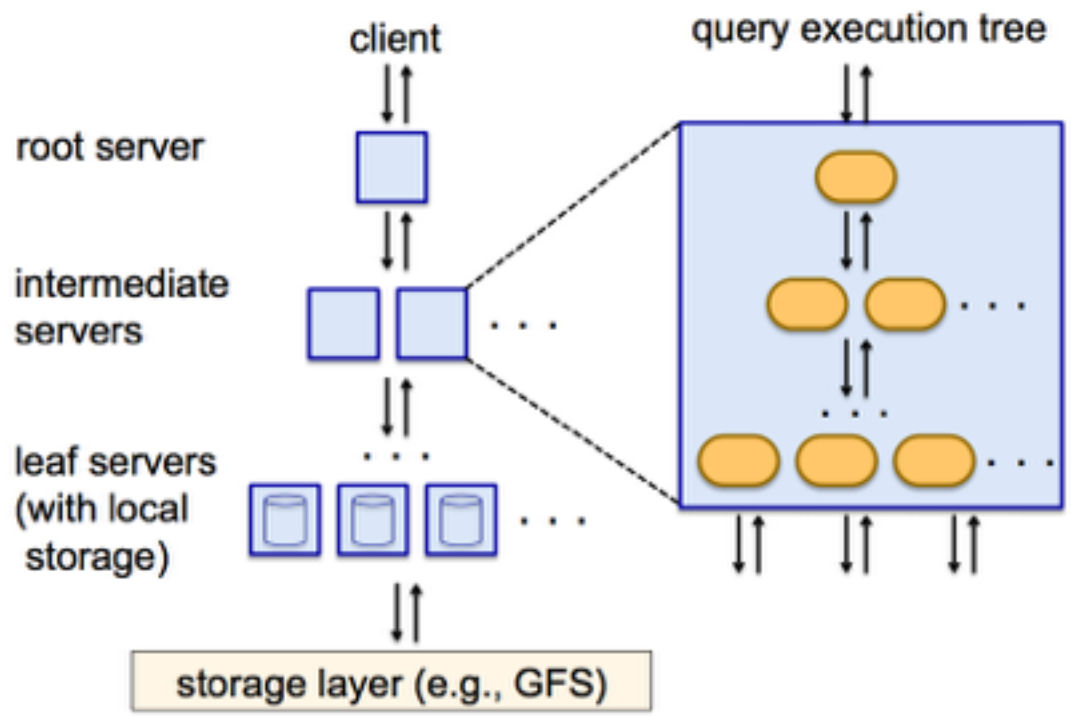
Routes the queries to the next level in the serving tree

SELECT A, COUNT(B) FROM T GROUP BY A

SELECT A, SUM(c) FROM (R_1 UNION ALL ... R_n^1) GROUP BY A

Query Execution

Dispatcher

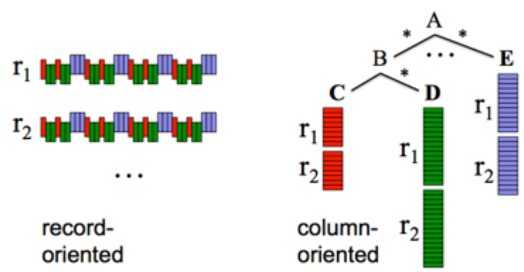


The amount of data processed in each query is often larger than the number of processing units available for execution, which we call slots.

3,000 leaf servers each using 8 threads has 24,000 slots

The query dispatcher honors a parameter that specifies the minimum percentage of tablets that must be scanned before returning a result

Data Model



DocId

value	r	d
10	0	0
20	0	0

Name.Url

value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

Links.Forward

value	r	d
20	0	2
40	1	2
60	1	2
80	0	2

Links.Backward

value	r	d
NULL	0	1
10	0	2
30	1	2

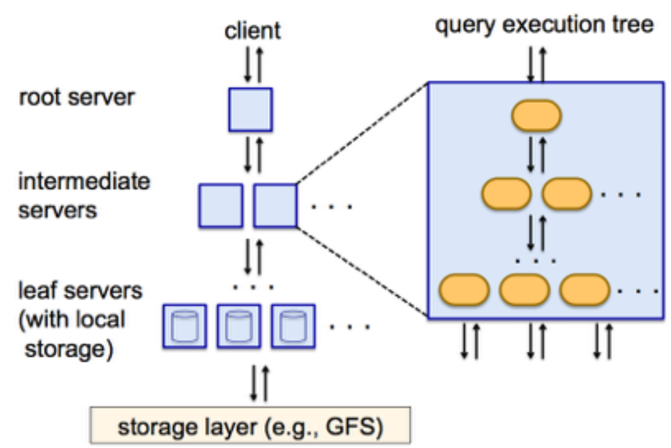
Name.Language.Code

value	r	d
en-us	0	2
en	2	2
NULL	1	1
en-gb	1	2
NULL	0	1

Name.Language.Country

value	r	d
us	0	3
NULL	2	2
NULL	1	1
gb	1	3
NULL	0	1

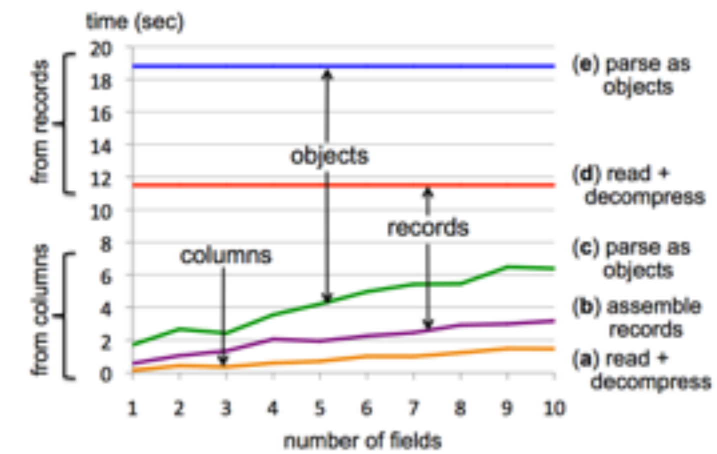
Query

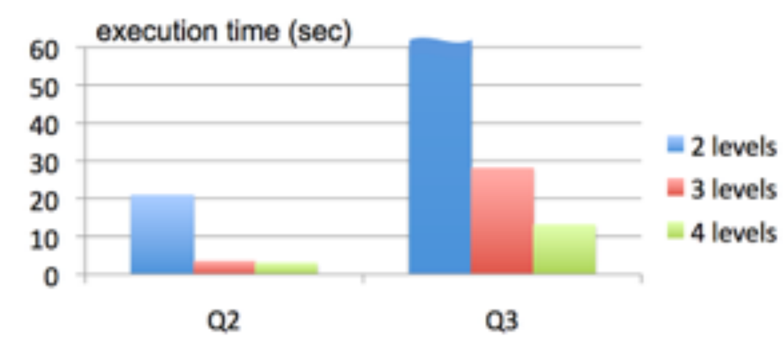


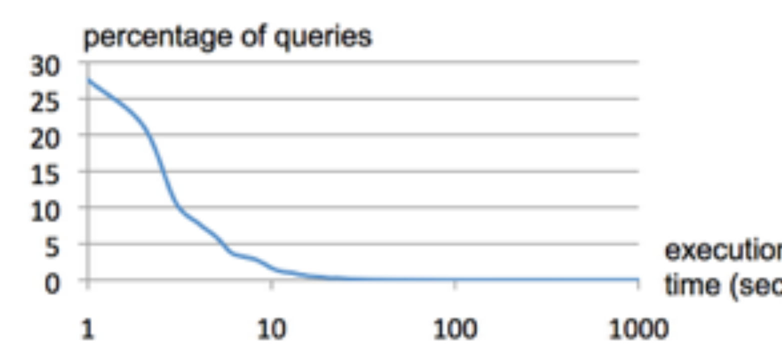
SELECT A, COUNT(B) FROM T GROUP BY A

SELECT A, SUM(c) FROM (R_1 UNION ALL ... R_n^1) GROUP BY A

Experiments







Experiments

Table name	Number of records	Size (unrepl., compressed)	Number of fields	Data center	Repl. factor
T1	85 billion	87 TB	270	A	3×
T2	24 billion	13 TB	530	A	3×
T3	4 billion	70 TB	1200	A	3×
T4	1+ trillion	105 TB	50	B	3×
T5	1+ trillion	20 TB	30	B	2×

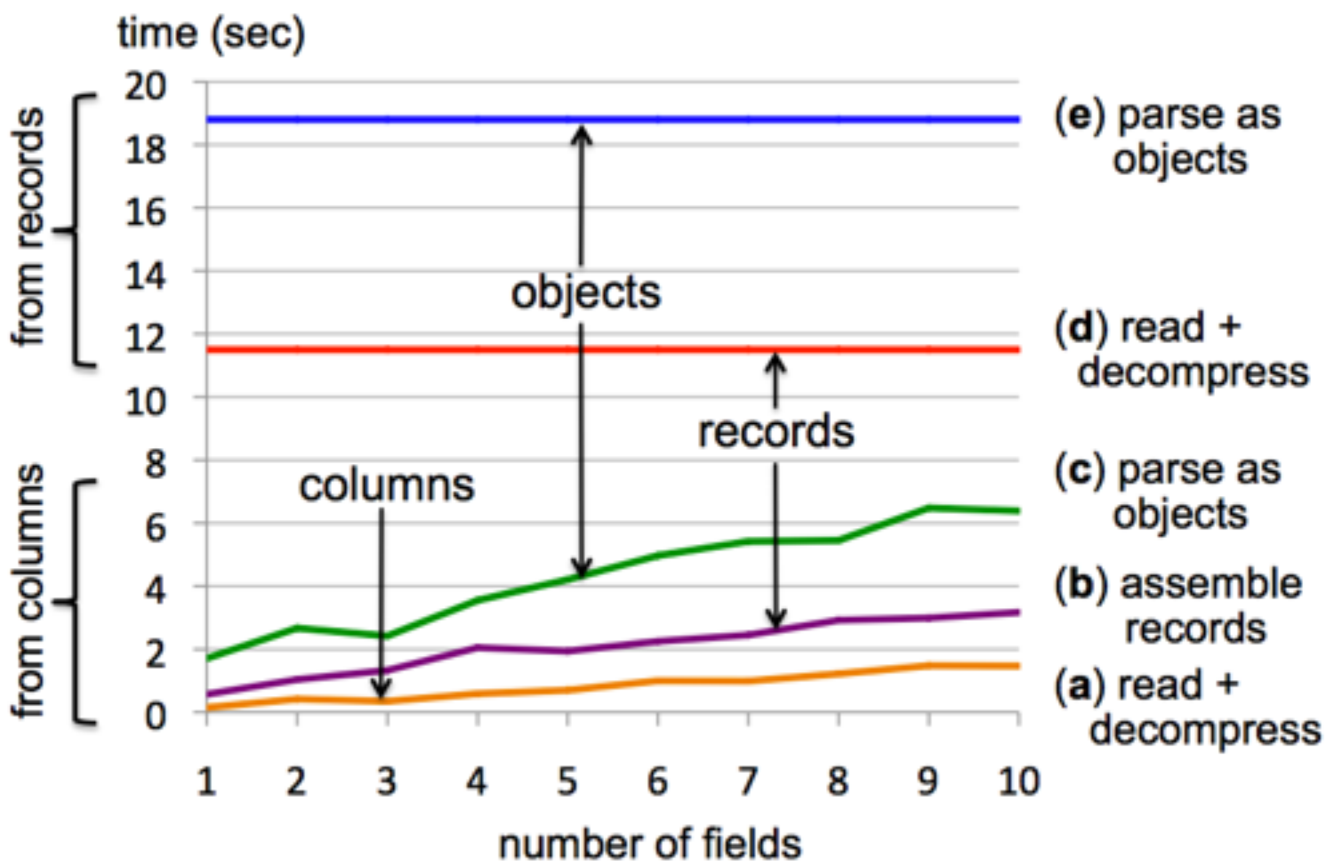
Local Disk   MR & Dremel   Scalability

Pre-tablet Histogram   Serving Tree Topology

Within-Record Aggregation   Straggler

Experiments

Local Disk



Tradeoffs of columnar vs. record-oriented storage

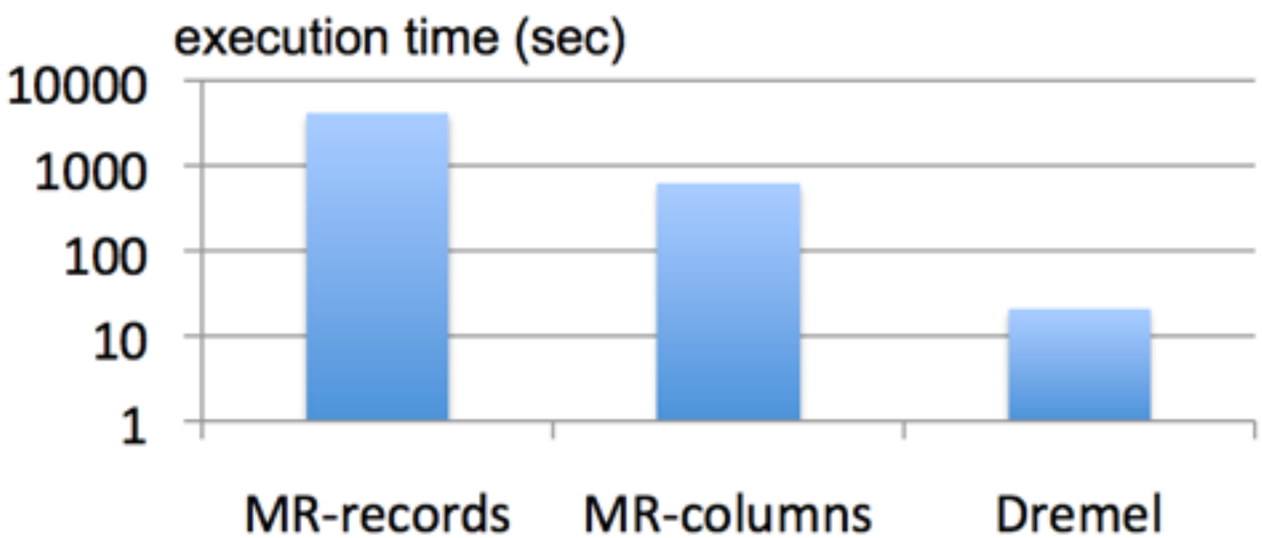
Record assembly and parsing are expensive, each potentially doubling the execution time.

A bulk of the time is spent in decompression; The compressed data can be read from the disk in about half the time.

Columnar Storage easier to compress

Experiments

MR & Dremel



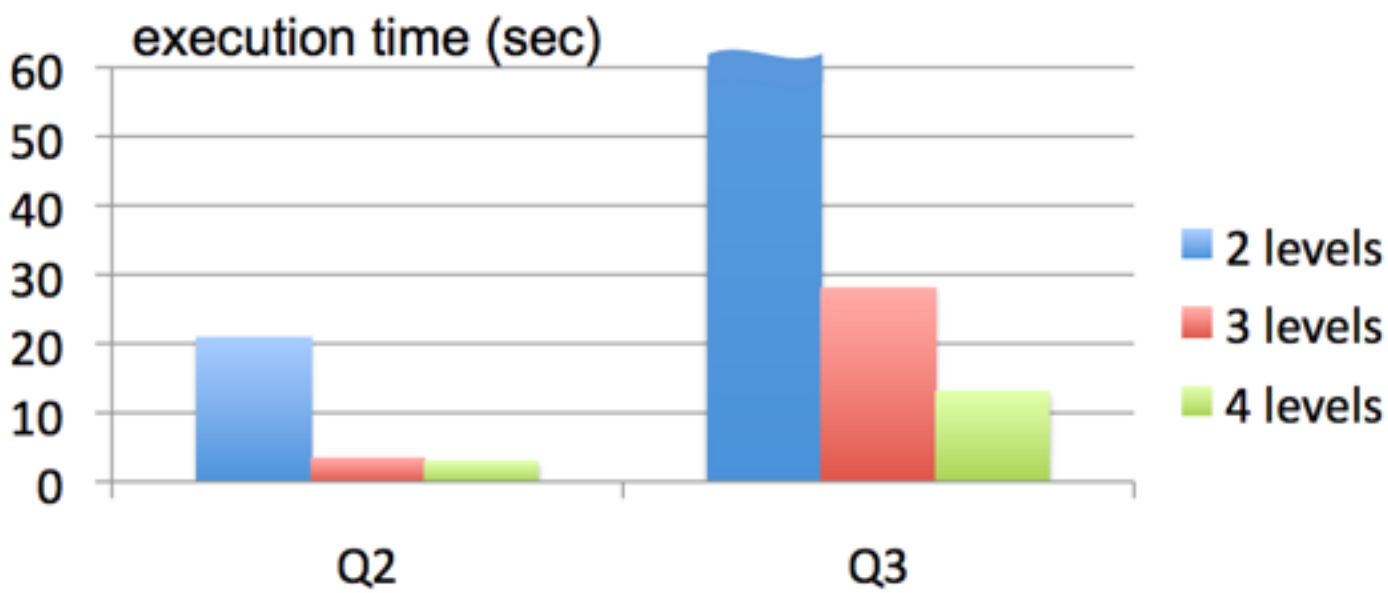
Q1: SELECT SUM(CountWords(txtField)) / COUNT(*)

FROM T1

Both MR jobs are run on 3000 workers.
Similarly, a 3000-node Dremel instance is used to execute Query Q1.
Dremel and MR-on-columns read about 0.5TB of compressed columnar data vs. 87TB (MR-on-records)

Experiments

Serving tree Topology



2 levels: 1:2900

3 levels: 1:100:2900

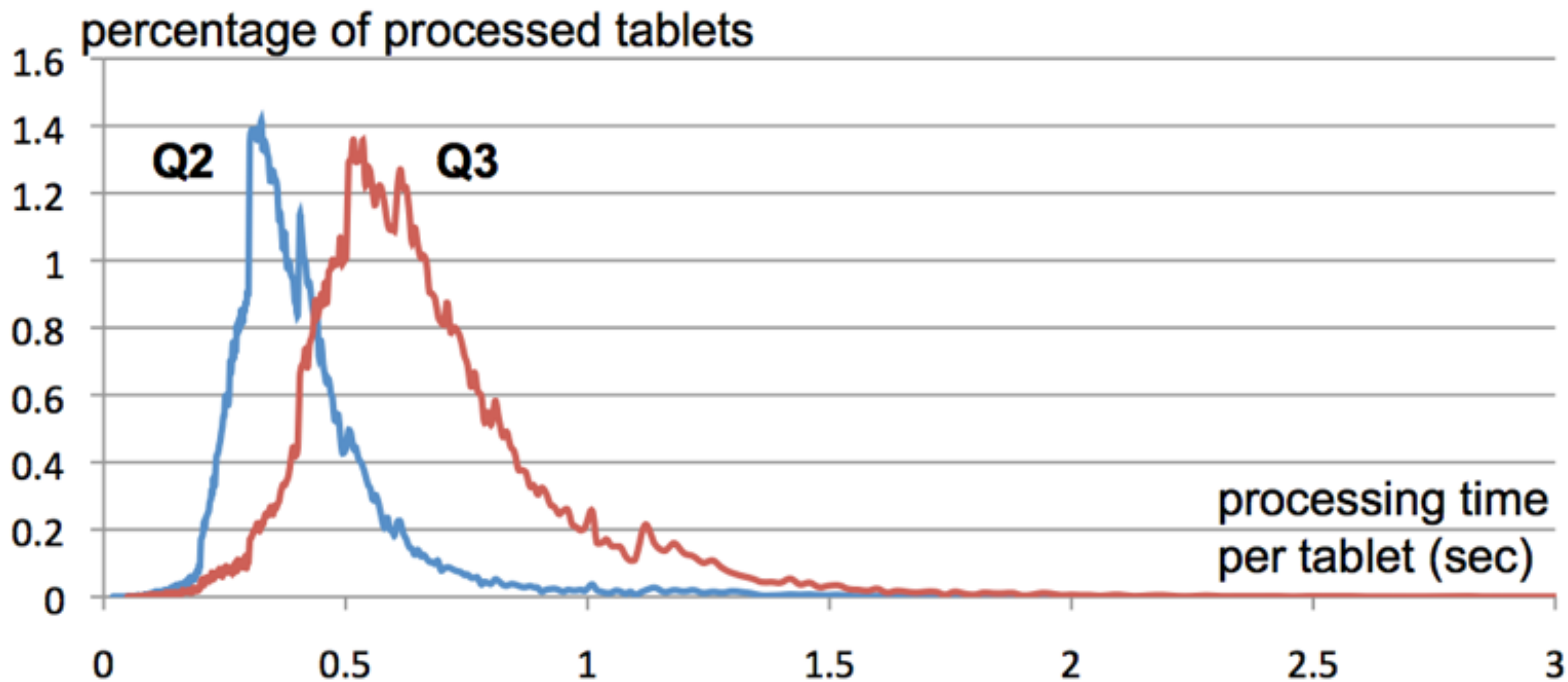
4 levels: 1:10:100:2900

Q2: SELECT country, SUM(item.amount) FROM T2 GROUP BY country

Q3: SELECT domain, SUM(item.amount) FROM T2 WHERE domain CONTAINS ’.net’
GROUP BY domain

Experiments

Per-tablet Histograms



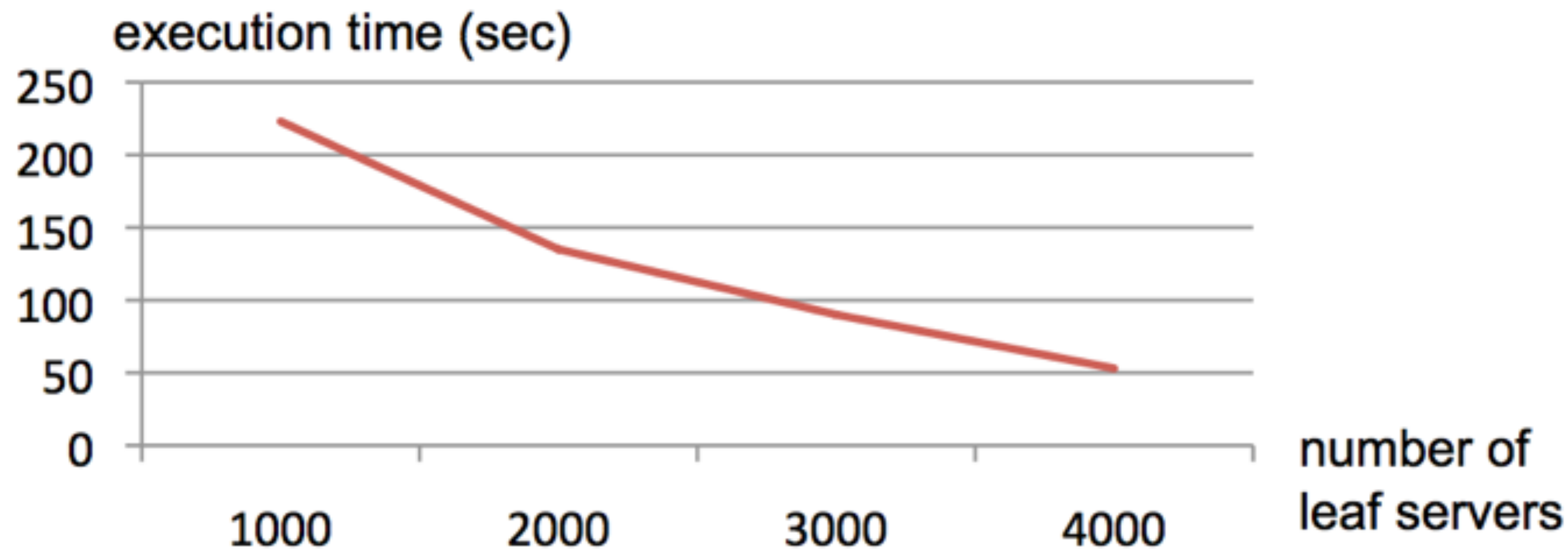
The figure shows how fast tablets get processed by the leaf servers for a specific run of Q2 and Q3.

Q2: SELECT country, SUM(item.amount) FROM T2 GROUP BY country

Q3: SELECT domain, SUM(item.amount) FROM T2 WHERE domain CONTAINS '.net' GROUP BY domain

Experiments

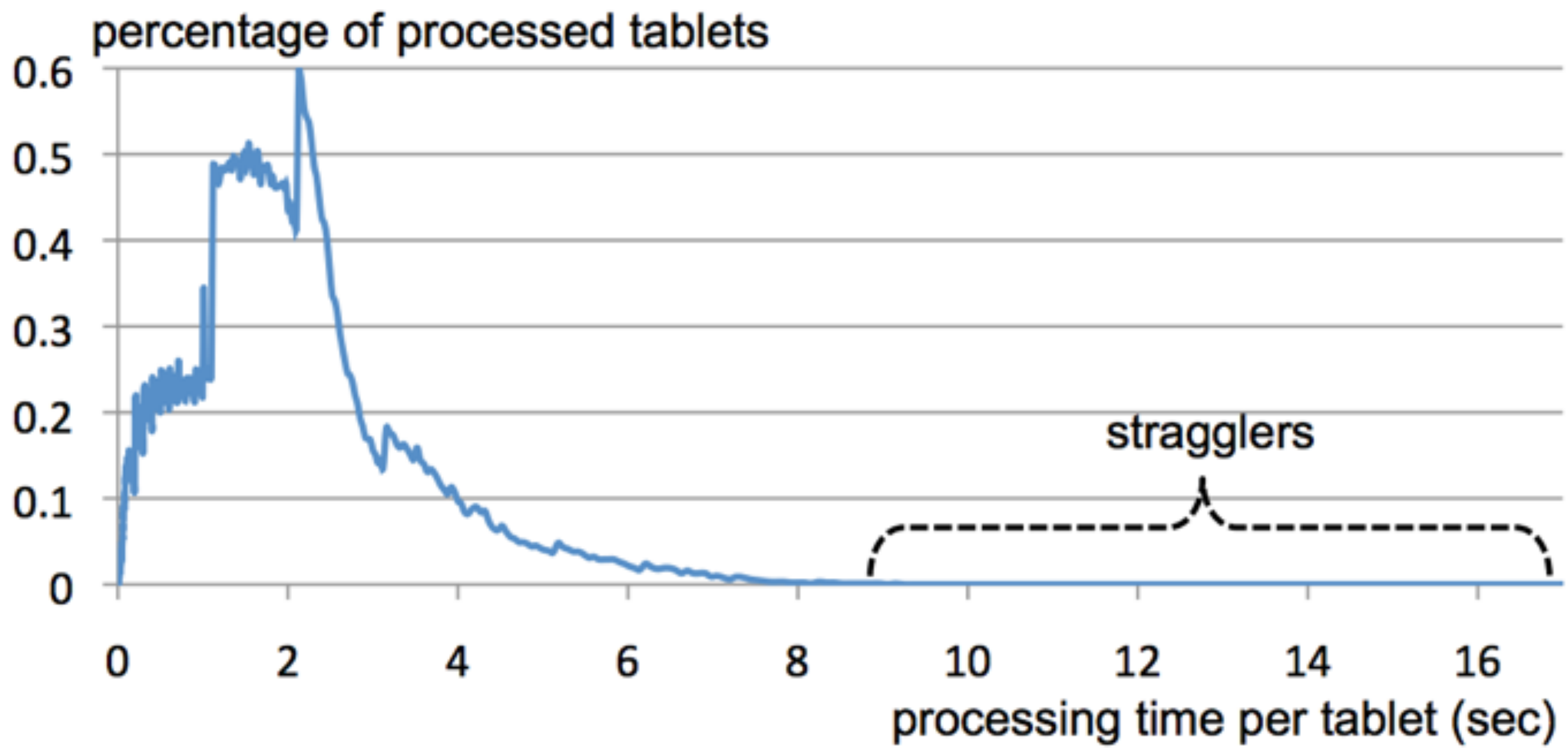
Scalability



Near-linear scalability in the number of columns and servers is achievable for systems containing thousands of nodes

Experiments

Stragglers



The processing time for 99% of the tablets is below 5 seconds per tablet per slot.

The Query is executed on a 2500 node system.

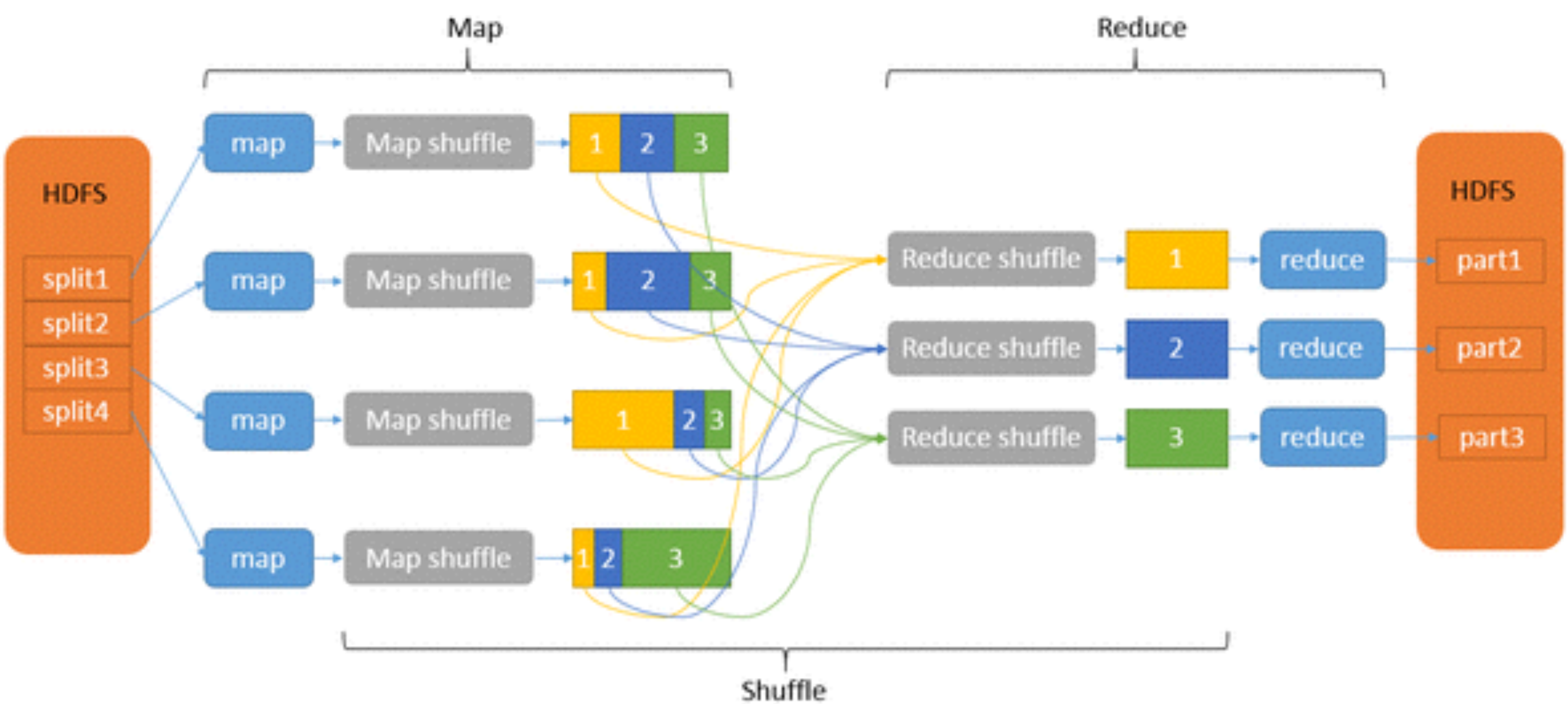
Summary

1. Scan-based queries can be executed at interactive speeds on disk-resident datasets of up to a trillion records.

2. In a multi-user environment,a larger system can benefit from economies of scale while offering a qualitatively better user experience.

3. If trading speed against accuracy is acceptable, a query can be terminated much earlier and yet see most of the data.
4. Column Store is naturally suitable for statistical analysis of data.

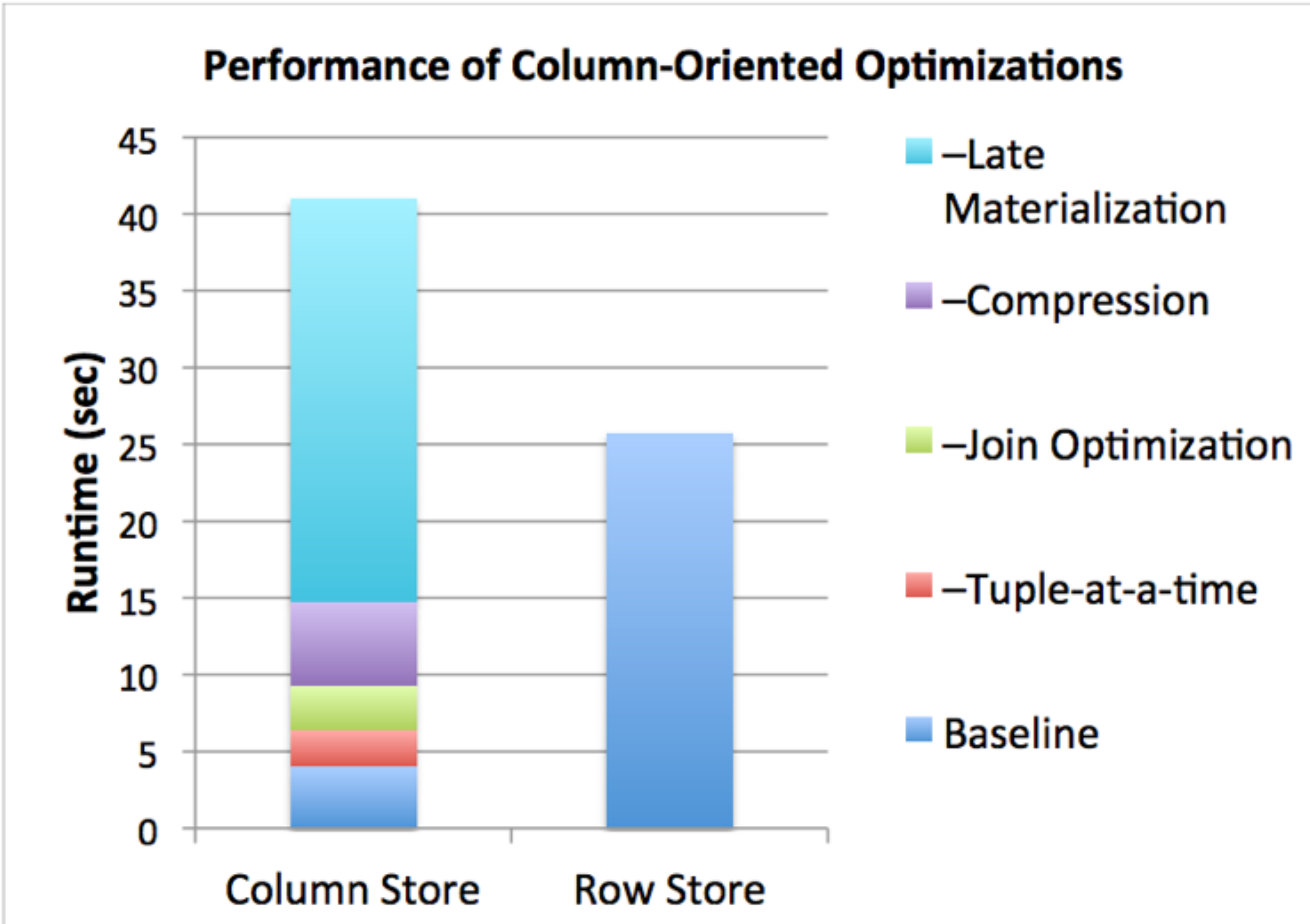
MapReduce



https://www.zhihu.com/question/23345991

Batch Mode

Reflection



The Design and Implementation of Modern Column-Oriented Database Systems

Reflection

DataFlow System

Data Model and Interface

Execution Engine

Scheduling

Fault Tolerance

Storage Layer

Reflection

DataFlow System

The need to reduce the gap between the generation of data and the generation of analytics results over this data has led to systems that can support both OLTP and OLAP workloads in a single system. — Massively Parallel Databases and MapReduce Systems

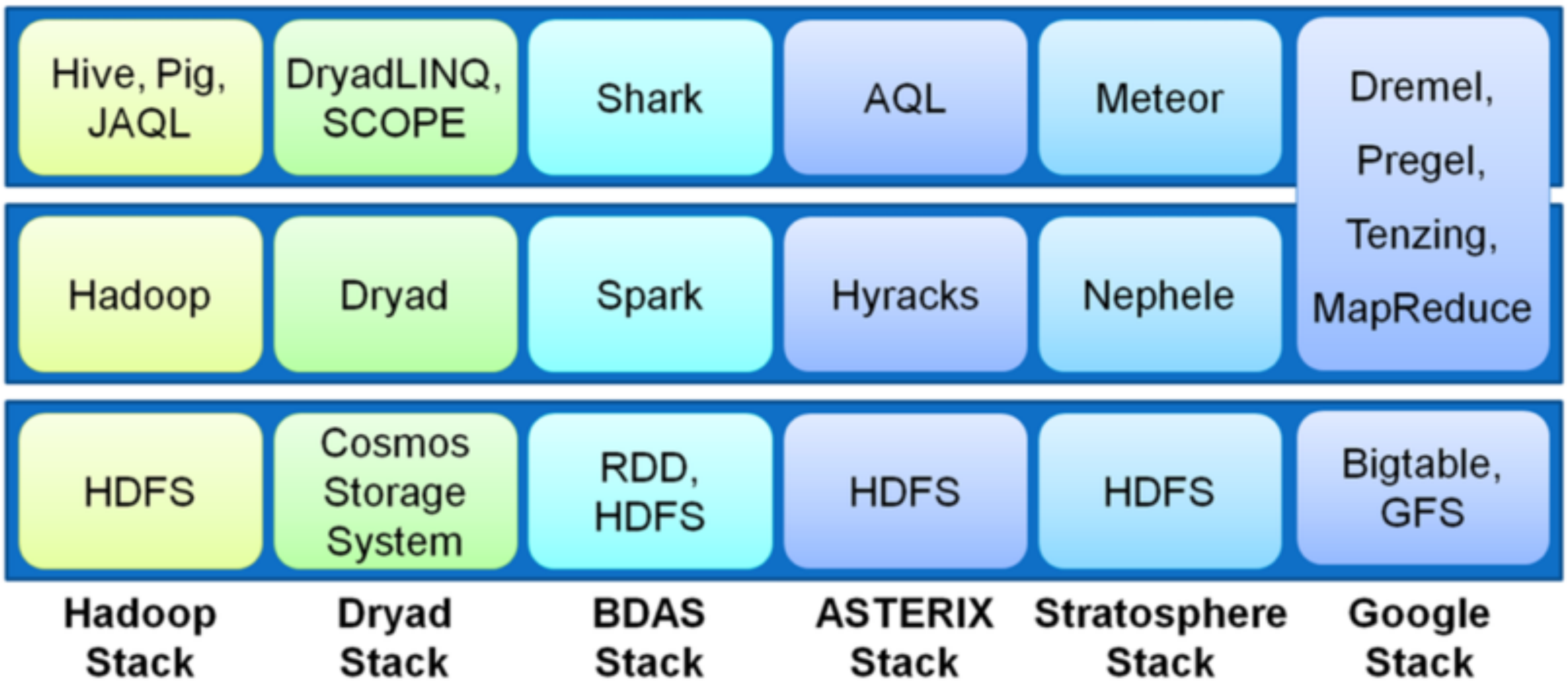
Whether could distribute the computing to Leaf Servers?

Data Format

Analytic Algorithms

Reflection

Dataflow Stack



Massively Parallel Databases and MapReduce Systems

Thank you!

Open source Implementation of Dremel

Cloudera Impala

Apache Drill