Dremel: Interactive Analysis Of Web-Scale Datasets

Why we need Dremel

- Large-scale analytical data processing becomes widespread
- Perform interactive data analysis at scale demands a high degree of parallelism
- Interactive response time matters
- Web and scientific computing data is often nested

How we built Dremel

- The ideas are from web search and parallel DBMSs
- Serving tree architecture
- SQL-like language
- Column-striped storage representation

Outline

- Data Model
- Contributions
- Experiments
- Conclusion

Data model

$$\tau = \mathbf{dom} \mid \langle A_1 : \tau[*|?], \ldots, A_n : \tau[*|?] \rangle$$

Data Model

- Language independent and platform neutral
- Strong-typed nested records

```
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; }}
```

$\mathbf{r_1}$

```
DocId: 10
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'
```

Contributions

Nested columnar storage

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

```
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'
```

- Repetition level: at what repeated file in the field's path the value has repeated (0 denotes the start of a new record)
- Definition level: how many fields in path that could be undefined (because they are optional or repeated) are actually present.

Nested columnar storage

- Required: no change, same as parent
- Optional: increment definition level
- Repeated: increment both

Nested columnar storage

Document: R = 0, D = 0

- DocId: *required*
- Links: *optional* D = 1
  - Backward: *repeated* R = 1, D = 2
  - Forward: *repeated* R = 1, D = 2
- Name: *repeated* R = 1, D = 1
  - Language: *repeated* R = 2, D = 2
    - Code: *required*
    - Country: *optional* D = 3
  - Url: *optional* D = 2

Name.Url

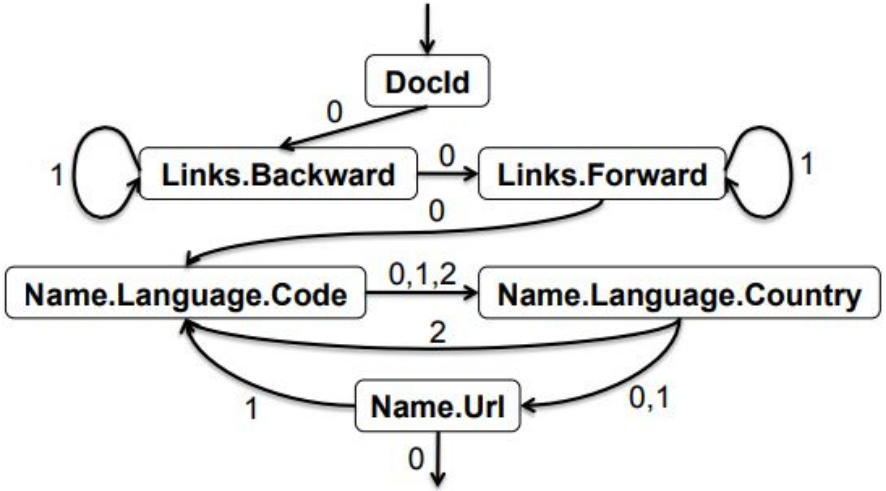
value	r	d
http://A	0	2
http://B	1	2
NULL	1	1
http://C	0	2

r_1

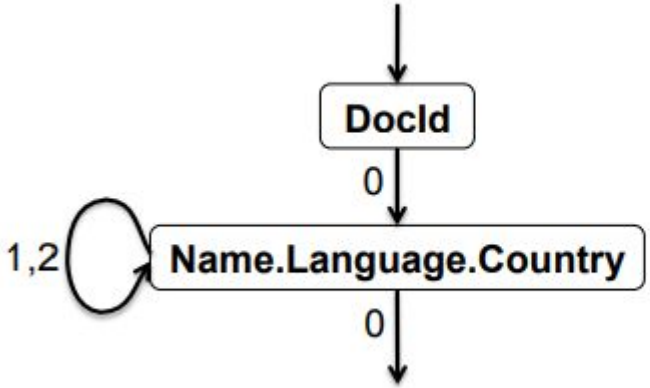
```
DocId: 10
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

Record assembly

- Create a finite state machine
- State transaction is labeled with repetition levels



Record assembly




```
DocId: 10                S1
Name
  Language
    Country: 'us'
  Language
Name
  Language
    Country: 'gb'
```

```
DocId: 20                S2
Name
```

```sql
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;
```

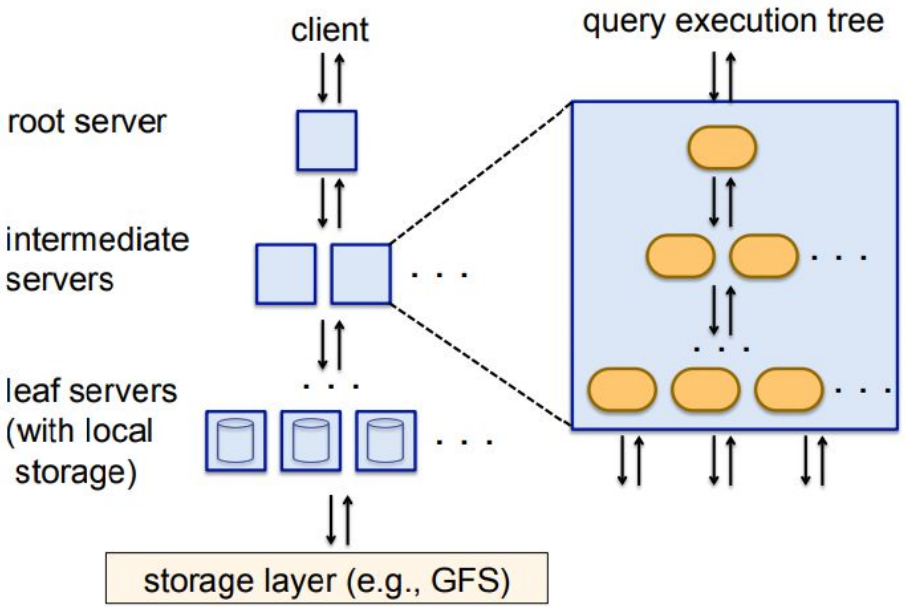
Query language

- Based on SQL
- Implemented on columnar nested storage
- Support nested subqueries, inter and intra-record aggregation, top-k, joins, user-defined functions, etc

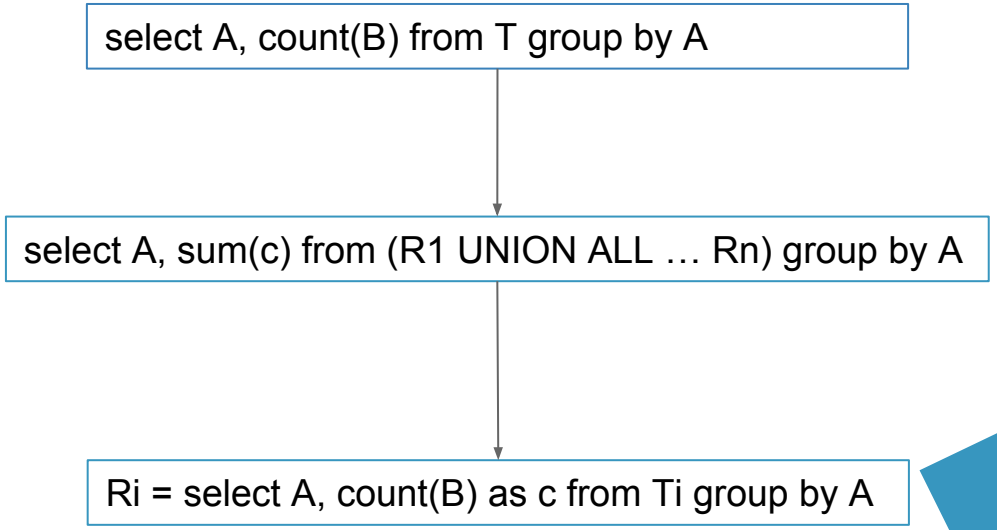
Query execution

Tree architecture

multi-level serving tree



Query execution

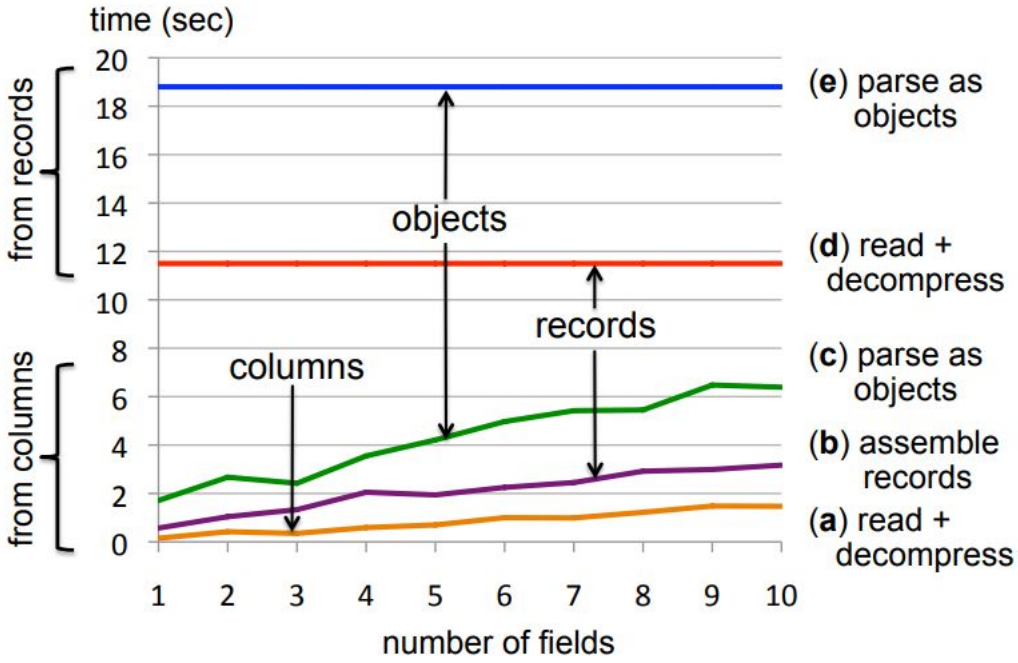


Dispatcher

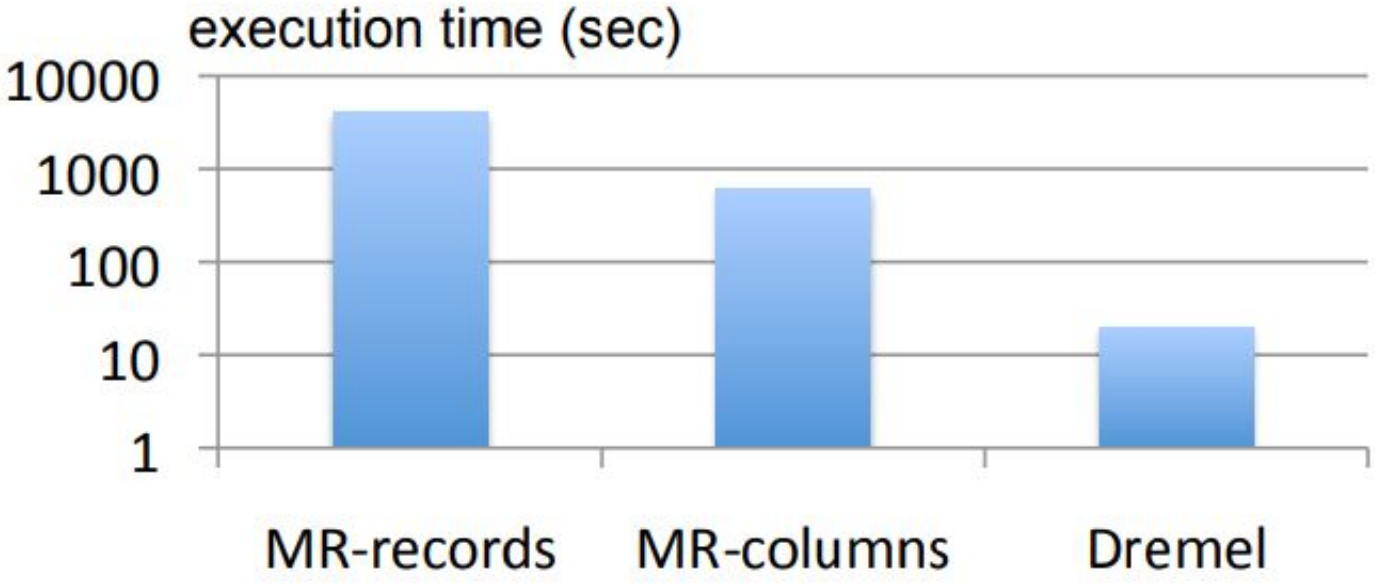
- Schedule queries
- Balance the load
- Provide fault tolerance

Experiments

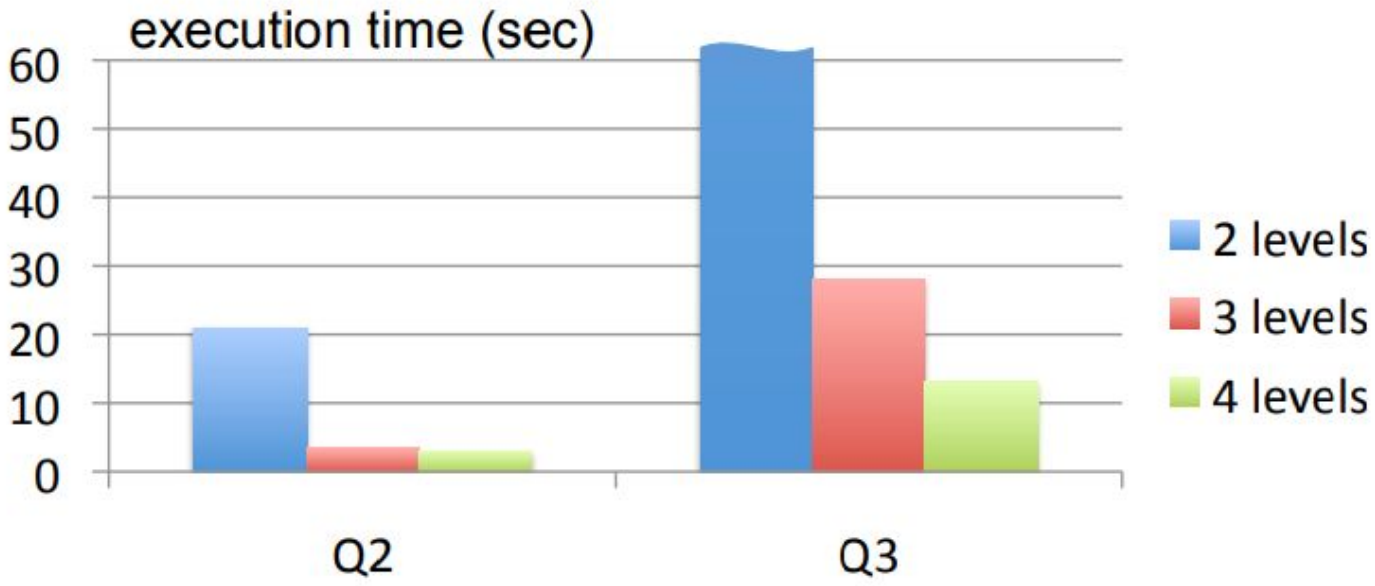
Experiment - local disk



Experiment - MR and Dremel



Experiment - Serving tree topology

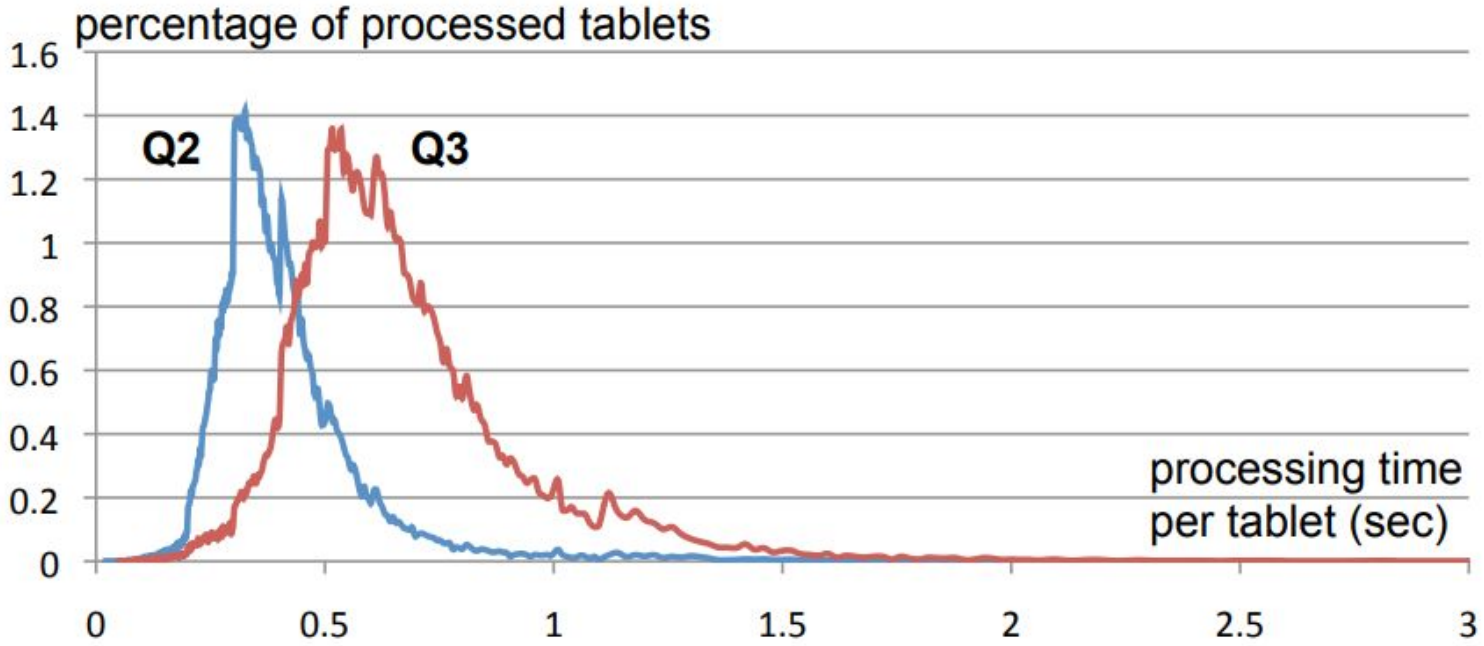


Experiment - within-record aggregation

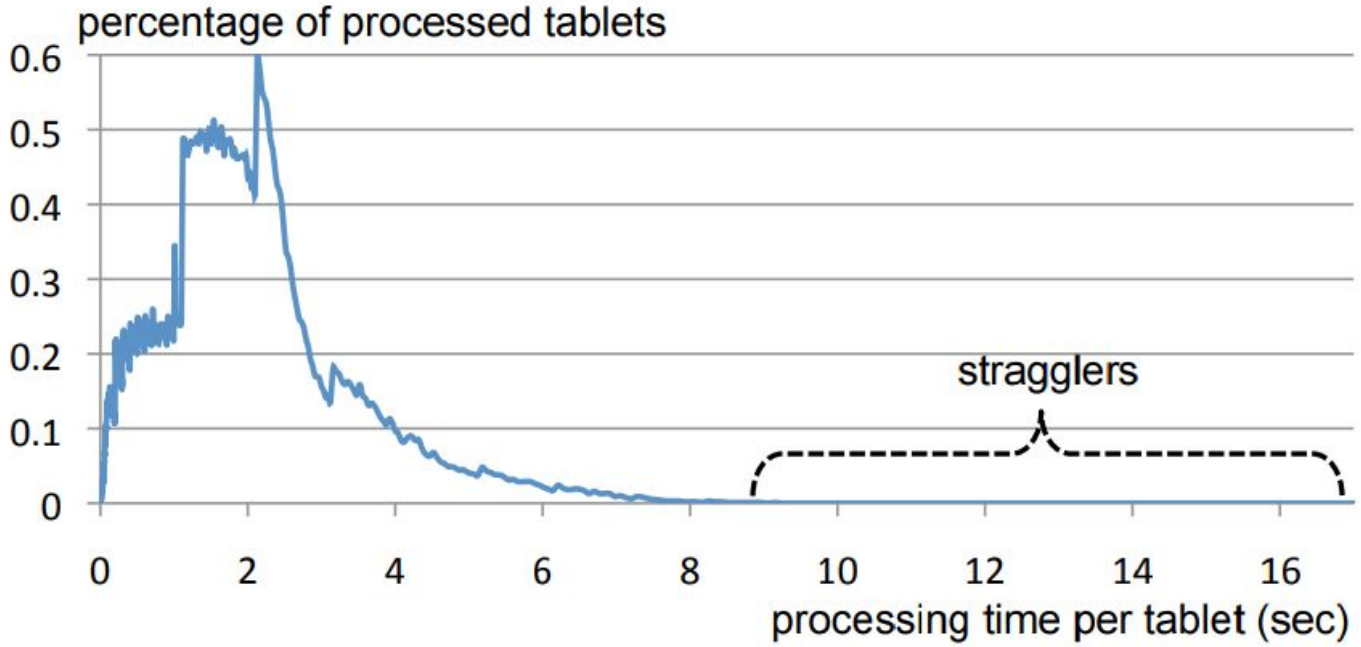
```
SELECT COUNT(c1 > c2) FROM
  (SELECT SUM(a.b.c.d) WITHIN RECORD AS c1,
          SUM(a.b.p.q.r) WITHIN RECORD AS c2
          FROM T3)
```

15 seconds !!!

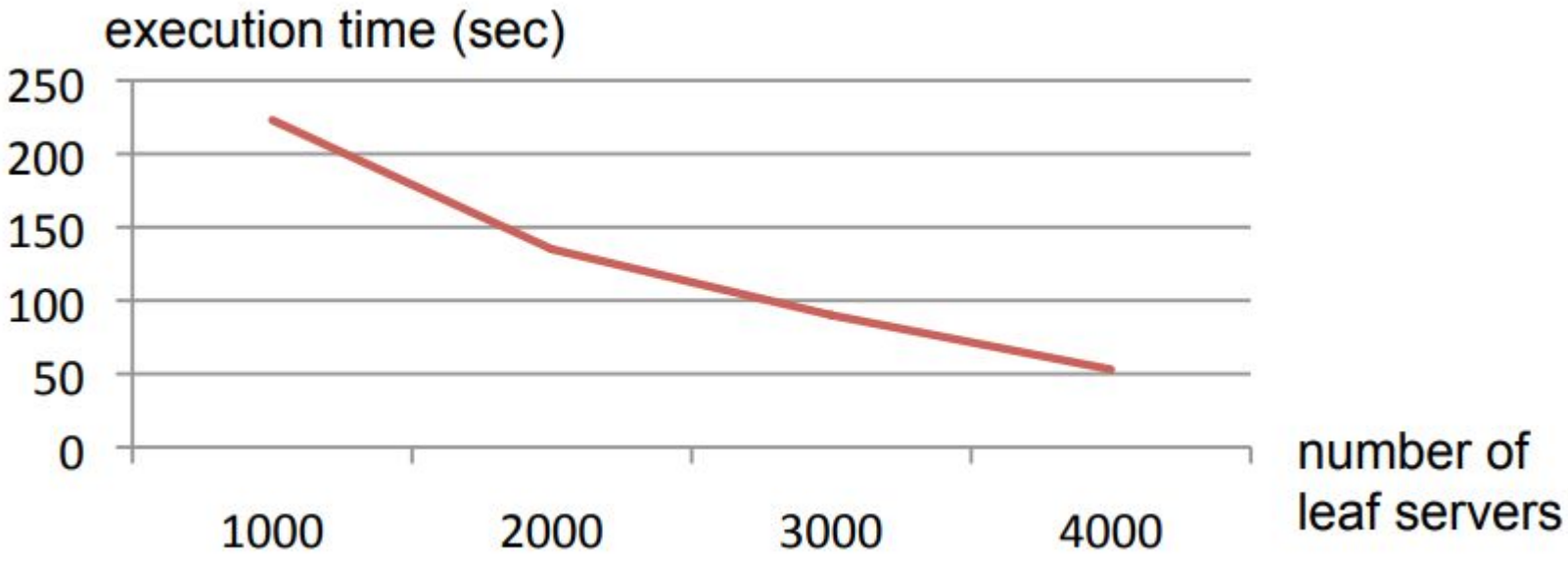
Experiment - pre-table histograms



Experiment - stragglers



Experiment - Scalability



Conclusion

Conclusion

	Map Reduce	Dremel
Data processing	Record Oriented	Column Oriented
In-situ processing	No	Yes
Size of queries	Large	Small/medium

Conclusion

Multi-level execution tree

+

Columnar data layout

→

Scalable & Efficient

Question?

Reference

- https://github.com/julienledem/redelm/wiki/The-striping-and-assembly-algorithms-from-the-Dremel-paper
- https://blog.twitter.com/engineering/en_us/a/2013/dremel-made-simple-with-parquet.html