Scalable PostgreSQL as your data platform

Ben Redman – ben@citusdata.com

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
    - HSTORE – semi-structured data in your DB
    - HLL – distinct counts using mathematical magic
    - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

citusdata

A Data Platform solves Data Problems

citusdata

Data Problems

- LOTS of data
- Changing needs
- Need a way to work with data I understand

Data Platform

- Store and query ALL your data
- Scalable
- Cost effective
- Extensible

citusdata

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
    - HSTORE – semi-structured data in your DB
    - HLL – distinct counts using mathematical magic
    - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

PostgreSQL

- http://www.postgresql.org/

- Used by:

SONY

NTT docomo

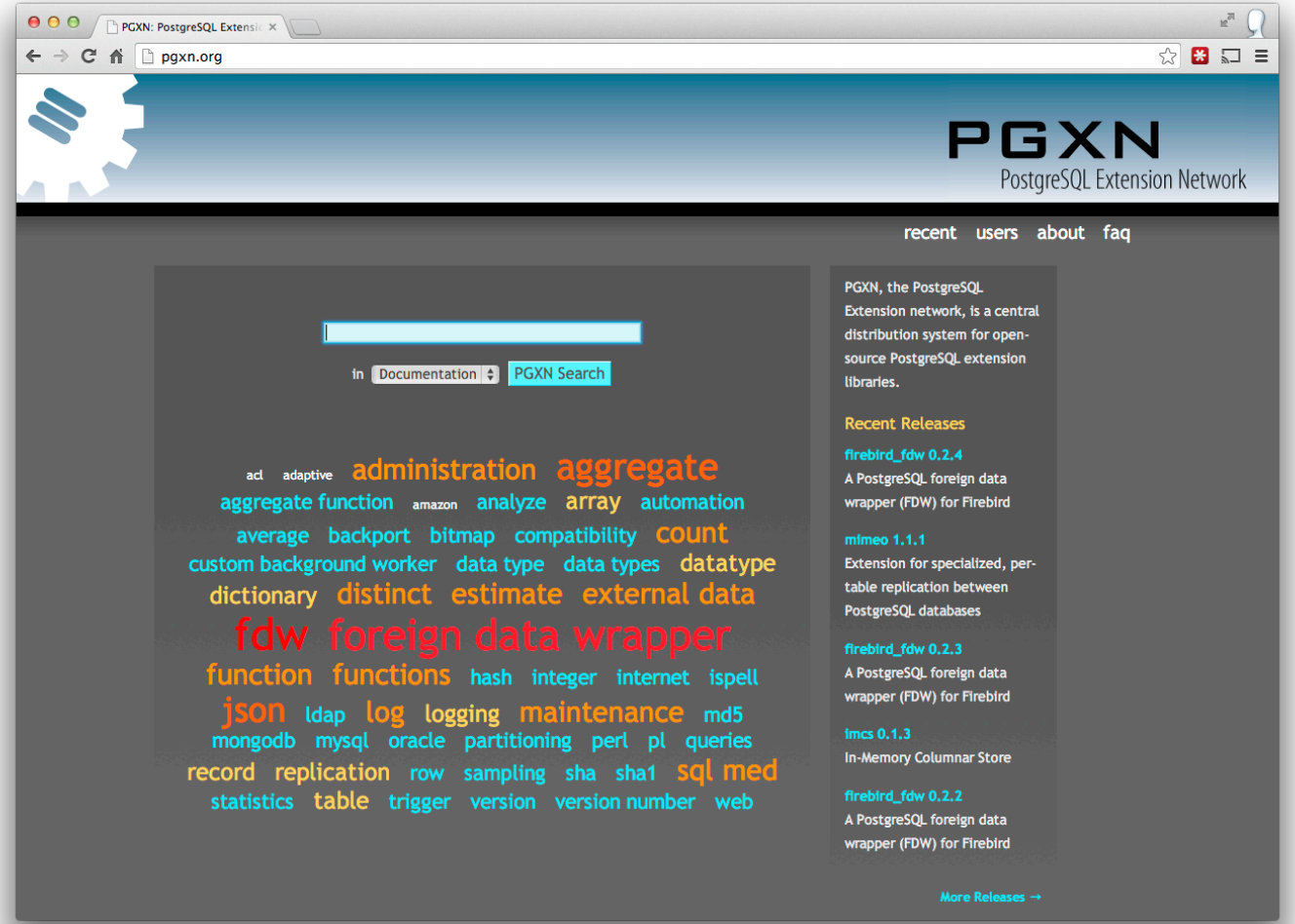
skype™

reddit

citusdata

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
   - HSTORE – semi-structured data in your DB
   - HLL – distinct counts using mathematical magic
   - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

citusdata


PGXN: PostgreSQL Extensi
pgxn.org
PGXN
PostgreSQL Extension Network
recent users about faq
in Documentation
PGXN Search
acl adaptive administration aggregate
aggregate function amazon analyze array automation
average backport bitmap compatibility count
custom background worker data type data types datatype
dictionary distinct estimate external data
fdw foreign data wrapper
function functions hash integer internet ispell
json ldap log logging maintenance md5
mongodb mysql oracle partitioning perl pl queries
record replication row sampling sha sha1 sql med
statistics table trigger version version number web
PGXN, the PostgreSQL Extension network, is a central distribution system for open-source PostgreSQL extension libraries.
Recent Releases
firebird_fdw 0.2.4
A PostgreSQL foreign data wrapper (FDW) for Firebird
mimeo 1.1.1
Extension for specialized, per-table replication between PostgreSQL databases
firebird_fdw 0.2.3
A PostgreSQL foreign data wrapper (FDW) for Firebird
imcs 0.1.3
In-Memory Columnar Store
firebird_fdw 0.2.2
A PostgreSQL foreign data wrapper (FDW) for Firebird
More Releases →

citusdata

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
   - HSTORE – semi-structured data in your DB
   - HLL – distinct counts using mathematical magic
   - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

citusdata

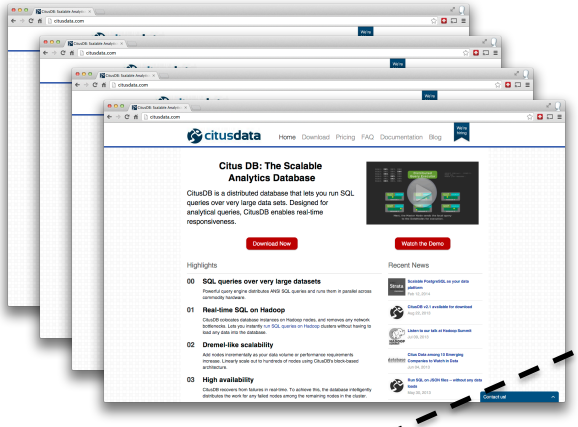
What is HSTORE?

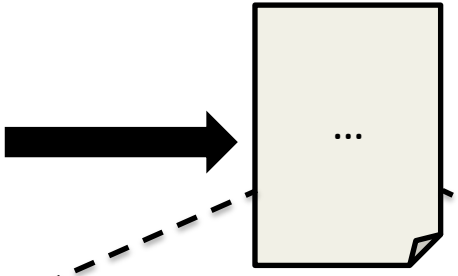
Data-type that enables storage of semi-structured data in key/value pairs.

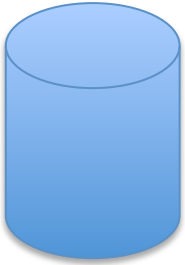
Compare to JSON.

```
{
   “referrer” => “…”,
   “ua” => “Mozilla/5…”,
   “cc” => “max-age…”,
   “ae” => “UTF-8”,
   “host” => “WWW…”
}
```

citusdata

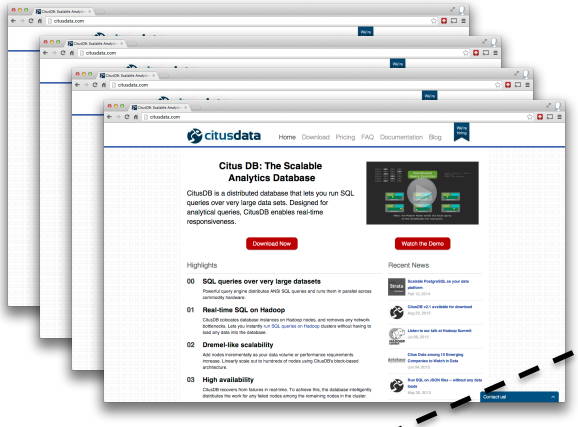


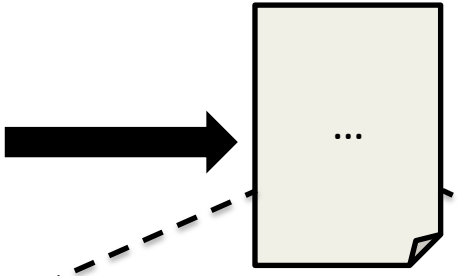


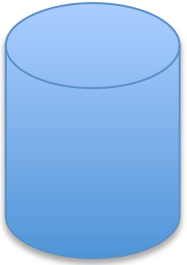


```
24.84.219.32
[10/Apr/2003:04:20:07 -0700]
"GET /archive/cat/games/index.shtml HTTP/1.1”
"http://www.google.ca/search?q=games+speed&ie=UTF-8&start=40"
"Mozilla/4.0 (MSIE 6.0; Windows 98; .NET CLR 1.0.3705)"
```

citusdata







```
24.84.219.32
[10/Apr/2003:04:20:07 -0700]
"GET /archive/cat/games/index.shtml HTTP/1.1”
"http://www.google.ca/search?q=games+speed&ie=UTF-8&start=40
"Mozilla/4.0 (MSIE 6.0; Windows 98; .NET CLR 1.0.3705)"
```

```
postgres=# CREATE EXTENSION hstore;
...
postgres=# \d access_logs
                    Table "public.access_logs"
```

Column	Type	Modifiers
time	timestamp without time zone	
request_ip	inet	
request_path	text	
referer_domain	text	
referer_path	text	
referer_query_params	hstore	

```
postgres=#
```

```
postgres=# select referer_query_params from access_logs where referer_domain
           LIKE '%google%’ LIMIT 5;
```

referer_query_params
"q"=>"cd+rom+drive+open+html", "hl"=>"en", "ie"=>"ISO-8859-1"
"q"=>"Heart+of+the+Alien", "hl"=>"it", "ie"=>"UTF-8", "oe"=>"UTF-8", "source
"q"=>"star+war+kid+download", "hl"=>"en", "ie"=>"ISO-8859-1", "lr"=>"", "saf
"q"=>"stupid+cubs+fan+video", "hl"=>"en", "ie"=>"UTF-8", "lr"=>"", "oe"=>"UT
"q"=>"star+wars+kid", "hl"=>"en", "ie"=>"UTF-8", "oe"=>"UTF-8"

```
(5 rows)

postgres=#
```

citusdata

```
postgres=# select count(*),
    CASE WHEN referer_query_params->'q' ~ 'star.*wars' THEN 'star wars'
    ELSE 'not star wars' END AS search_type
    FROM access_logs
    WHERE referer_domain='www.google.com’ AND referer_query_params ? 'q'
    GROUP BY search_type;
```

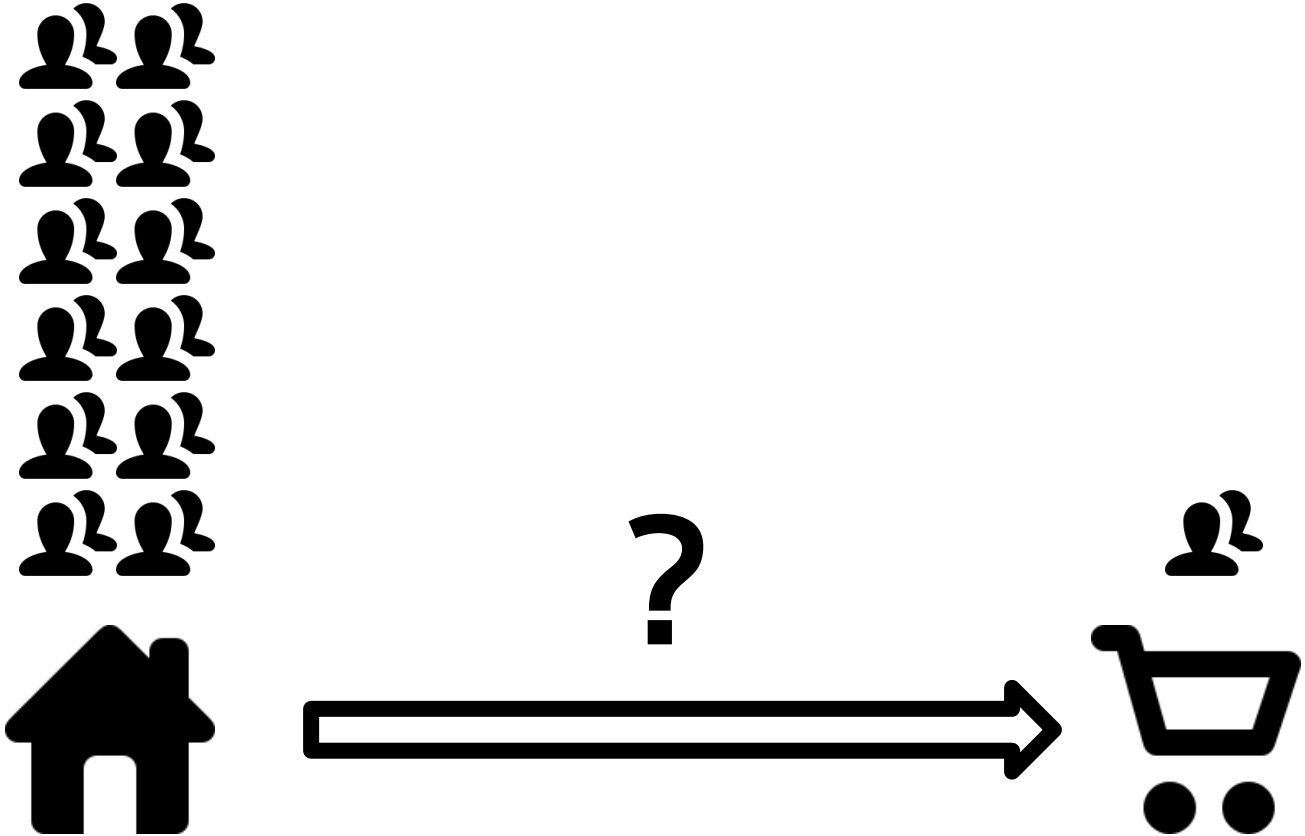
count	search_type
25316	star wars
38251	not star wars

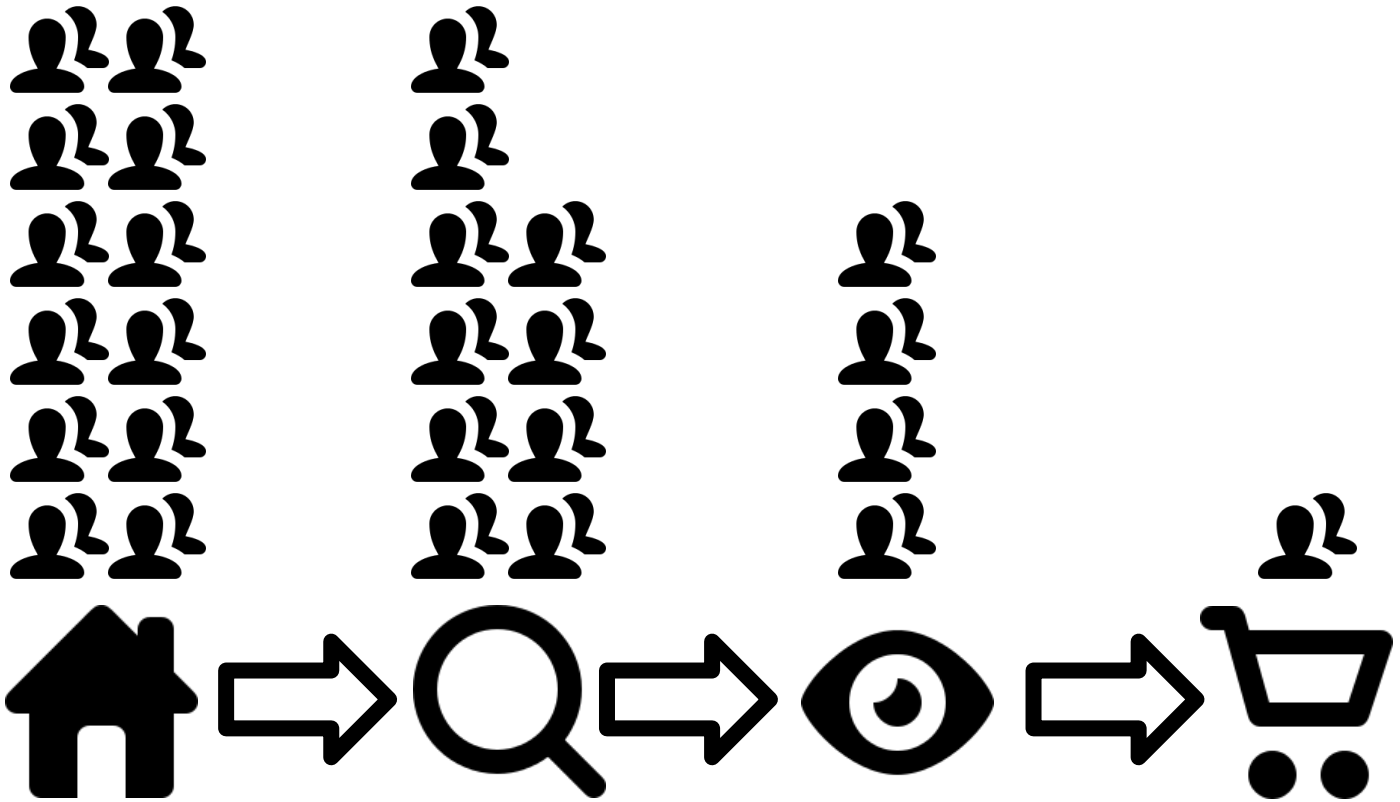
(2 rows)

postgres=#

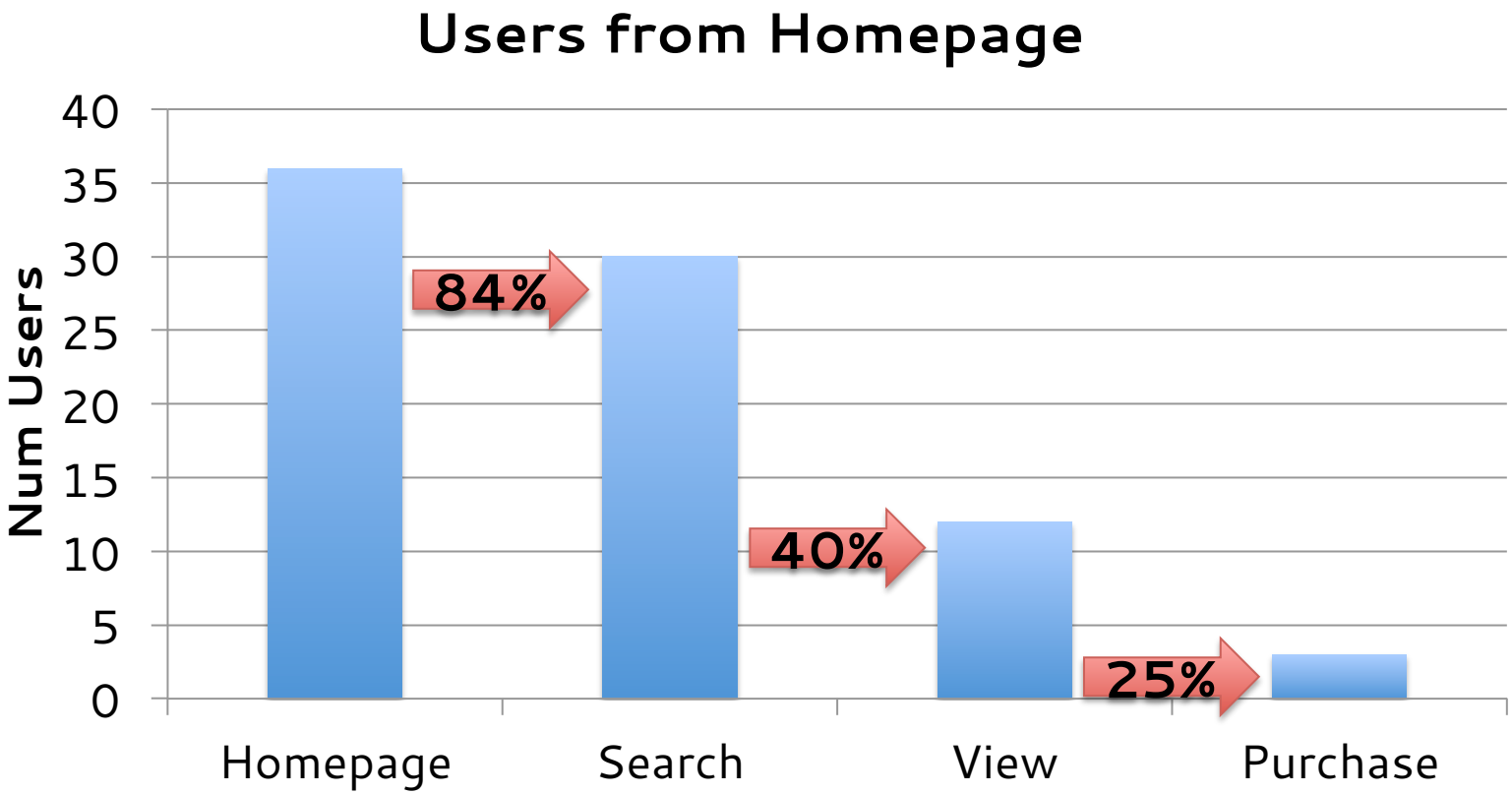
HSTORE goodness

- Support for indexing on HSTORE
- Still under development; coming soon nested HSTORE support



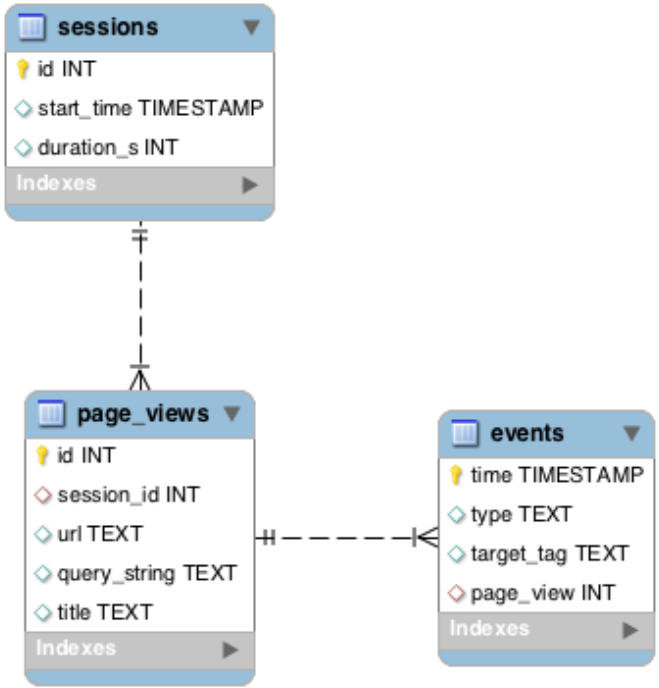


citusdata


Users from Homepage
Num Users
40
35
30
25
20
15
10
5
0
84%
40%
25%
Homepage
Search
View
Purchase

citusdata

Original Model



citusdata

```
WITH “login" AS (
    SELECT event.user_id, min(event.time) AS mintime
    FROM event_pruned "event"
    WHERE (("type" = 'click' AND ("target_tag" = 'input' AND "target_id" =
    'item_complete')) AND "app_id" = 3212870315 AND "event".app_id = 3212870315)
    GROUP BY event.user_id
),
“view_article" AS (
    SELECT event.user_id, min(event.time) AS mintime
    FROM event_pruned "event" NATURAL JOIN “login"
    WHERE (("type" = 'submit' AND ("target_class" ilike '%well%' AND "target_tag" =
    'form')) AND "app_id" = 3212870315 AND “login".user_id = "event".user_id AND
    “login".mintime < "event".time AND "event".app_id = 3212870315)
    GROUP BY event.user_id
),
"funnel" AS (
    SELECT *, CASE WHEN user_id IN (select user_id from “view_article") THEN 'View
    article' WHEN user_id IN (select user_id from “login") THEN 'Login' ELSE null END
    AS stage
    FROM "user"
    WHERE ("user".app_id = 3212870315)
)
SELECT count(*) AS count, stage FROM "funnel" WHERE ("funnel".app_id = 3212870315) GROUP BY
stage;
```

citusdata

New Model


```
events = [
   {
      “url” => “http://…”
      “query_string” => “…”
      “event_type” => “click”
      “event_tag” => “…”
   },
   {
 …
]
```

citusdata

Add new custom function

```
contains_elements(
 events [] hstore,
 search_sequence [] text
);
```

citusdata

```
contains_elements(
 [{ “event” => “home” }, { “event” => “find” },
  { “event” => “view” }, { “event” => “find” },
  { “event” => “find” }, { “event” => “rate” },
  { “event” => “home” }, { “event” => “add” },
  { “event” => “buy” }, { “event” => “view” },
 ],
 [ ‘event=>home’, ‘event=>find’, ‘event=>buy’ ]
);
```

citusdata

```
contains_elements(
  [{ “event” => “home” }, { “event” => “find” },
   { “event” => “view” }, { “event” => “find” },
   { “event” => “find” }, { “event” => “rate” },
   { “event” => “home” }, { “event” => “add” },
   { “event” => “buy” }, { “event” => “view” },
  ],
  [ ‘event=>home’, ‘event=>find’, ‘event=>buy’ ]
);
```

citusdata

```
contains_elements(
  [{ “event” => “home” }, { “event” => “find” },
   { “event” => “view” }, { “event” => “find” },
   { “event” => “find” }, { “event” => “rate” },
   { “event” => “home” }, { “event” => “add” },
   { “event” => “buy” }, { “event” => “view” },
  ],
  [ ‘event=>home’, ‘event=>find’, ‘event=>buy’ ]
);
```

citusdata

```
contains_elements(
  [{ “event” => “home” }, { “event” => “find” },
   { “event” => “view” }, { “event” => “find” },
   { “event” => “find” }, { “event” => “rate” },
   { “event” => “home” }, { “event” => “add” },
   { “event” => “buy” }, { “event” => “view” },
  ],
  [ ‘event=>home’, ‘event=>find’, ‘event=>buy’ ]
);
```

citusdata

New User Defined Function

`contains_elements(A hstore[], B text[])`

Returns 1 if there is a sub-sequence of A like $A[i_1]...A[i_{|B|}]$ such that $A[i_j]$ matches B[j] for all $1 <= j <= |B|$, otherwise it returns 0.

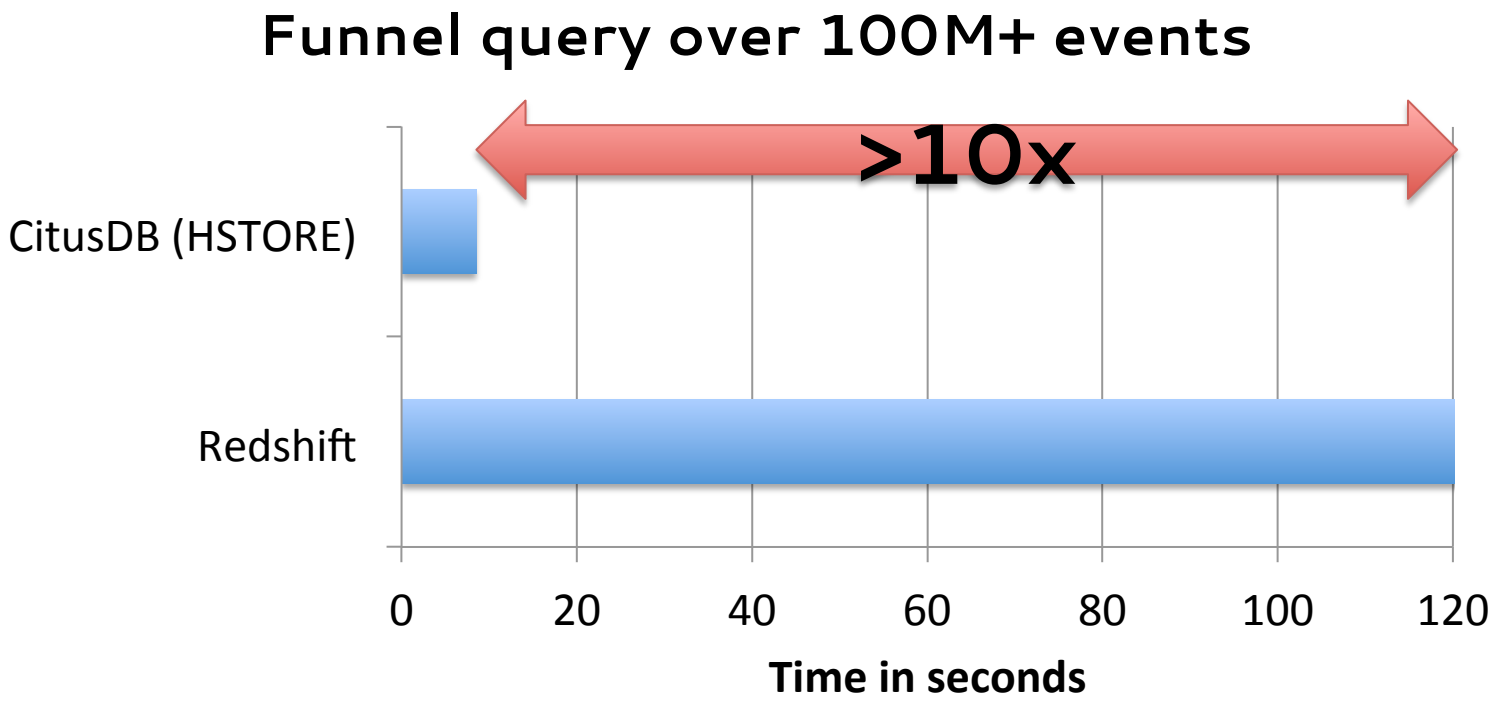
citusdata

```
WITH “login" AS (
    SELECT event.user_id, min(event.time) AS mintime
    FROM event_pruned "event"
    WHERE (("type" = 'click' AND ("target_tag" = 'input' AND "target_id" =
    'item_complete')) AND "app_id" = 3212870315 AND "event".app_id = 3212870315)
    GROUP BY event.user_id
),
“view_article" AS (
    SELECT event.user_id, min(event.time) AS mintime
    FROM event_pruned "event" NATURAL JOIN “login"
    WHERE (("type" = 'submit' AND ("target_class" ilike '%well%' AND "target_tag" =
    'form')) AND "app_id" = 3212870315 AND “login".user_id = "event".user_id AND
    “login".mintime < "event".time AND "event".app_id = 3212870315)
    GROUP BY event.user_id
),
"funnel" AS (
    SELECT *, CASE WHEN user_id IN (select user_id from “view_article") THEN 'View
    article' WHEN user_id IN (select user_id from “login") THEN 'Login' ELSE null END
    AS stage
    FROM "user"
    WHERE ("user".app_id = 3212870315)
)
SELECT count(*) AS count, stage FROM "funnel" WHERE ("funnel".app_id = 3212870315) GROUP BY
stage;
```

citusdata

```
SELECT
     sum(contains_elements(events, ARRAY['type=>home'])),
     sum(contains_elements(events, ARRAY['type=>home', 'type=>find'])),
     sum(contains_elements(events, ARRAY['type=>click', 'type=>find’, ‘type=>buy’]))
FROM
    event_sessions
WHERE
    app_id = 3212870315;
```

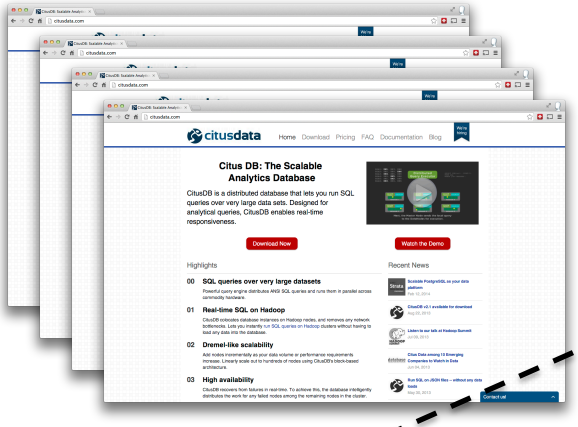
So what?

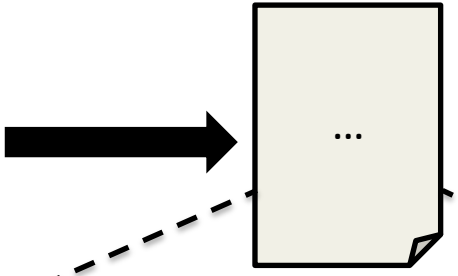


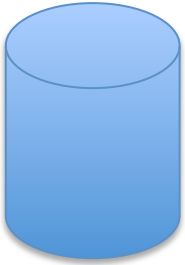
citusdata

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
    - HSTORE – semi-structured data in your DB
    - HLL – distinct counts using mathematical magic
    - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

citusdata





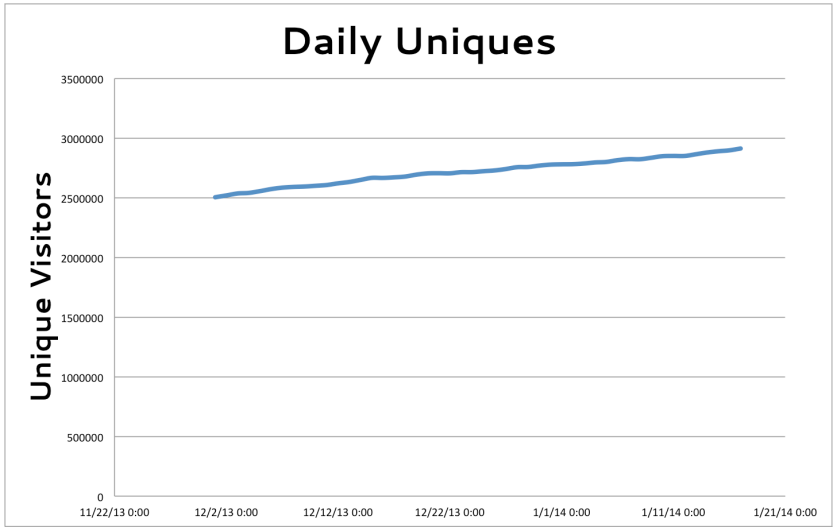


```
24.84.219.32
[10/Apr/2003:04:20:07 -0700]
"GET /archive/cat/games/index.shtml HTTP/1.1”
"http://www.google.ca/search?q=games+speed&ie=UTF-8&start=40"
"Mozilla/4.0 (MSIE 6.0; Windows 98; .NET CLR 1.0.3705)"
```

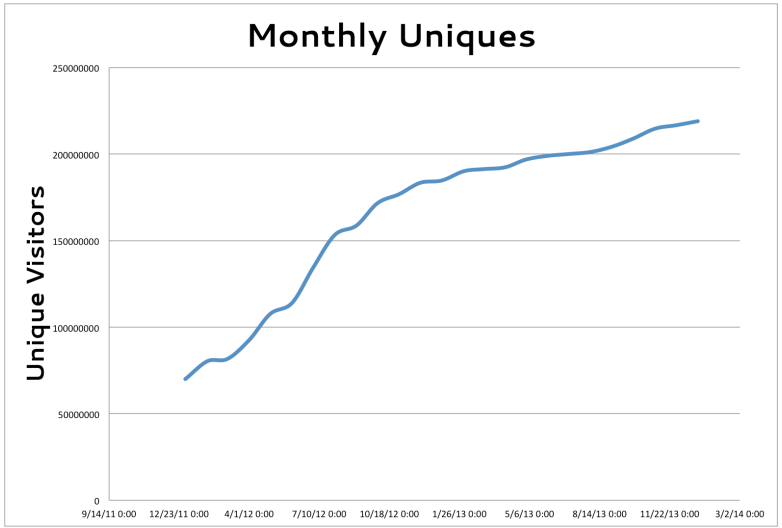
citusdata

Unique Viewer Counts

Daily data



Monthly data



citusdata

How to compute?

Time	IP	URL
1/2/14 12:03:04	87.32.182.4	/
1/2/14 12:03:04	243.32.93.2	/prod…
1/2/14 12:03:05	243.32.93.2	/prod…
…	…	…

```
SELECT
count(distinct(ip))
FROM requests
WHERE time >= '2014-01-01
12:00'
AND time < '2014-01-02
12:00';
```

citusdata

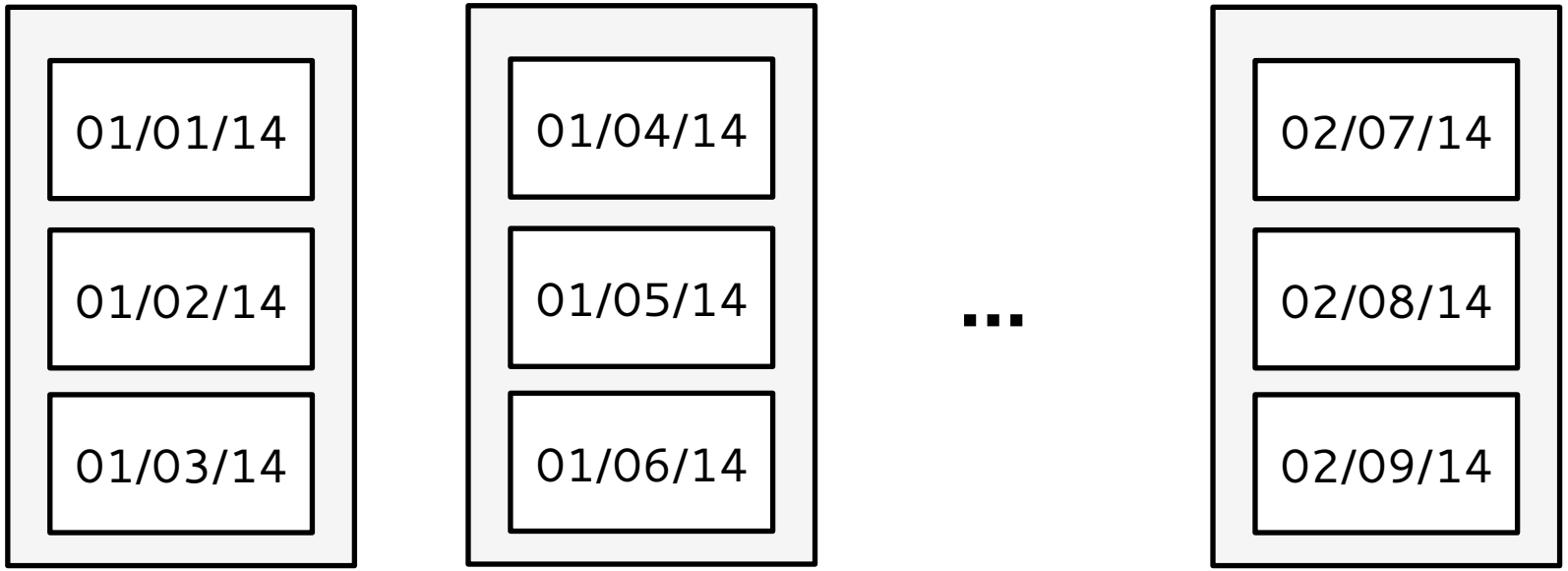
How to compute?

Time	IP	URL
1/2/14 12:03:04	87.32.182.4	/
1/2/14 12:03:04	243.32.93.2	/prod…
1/2/14 12:03:05	243.32.93.2	/prod…
…	…	…

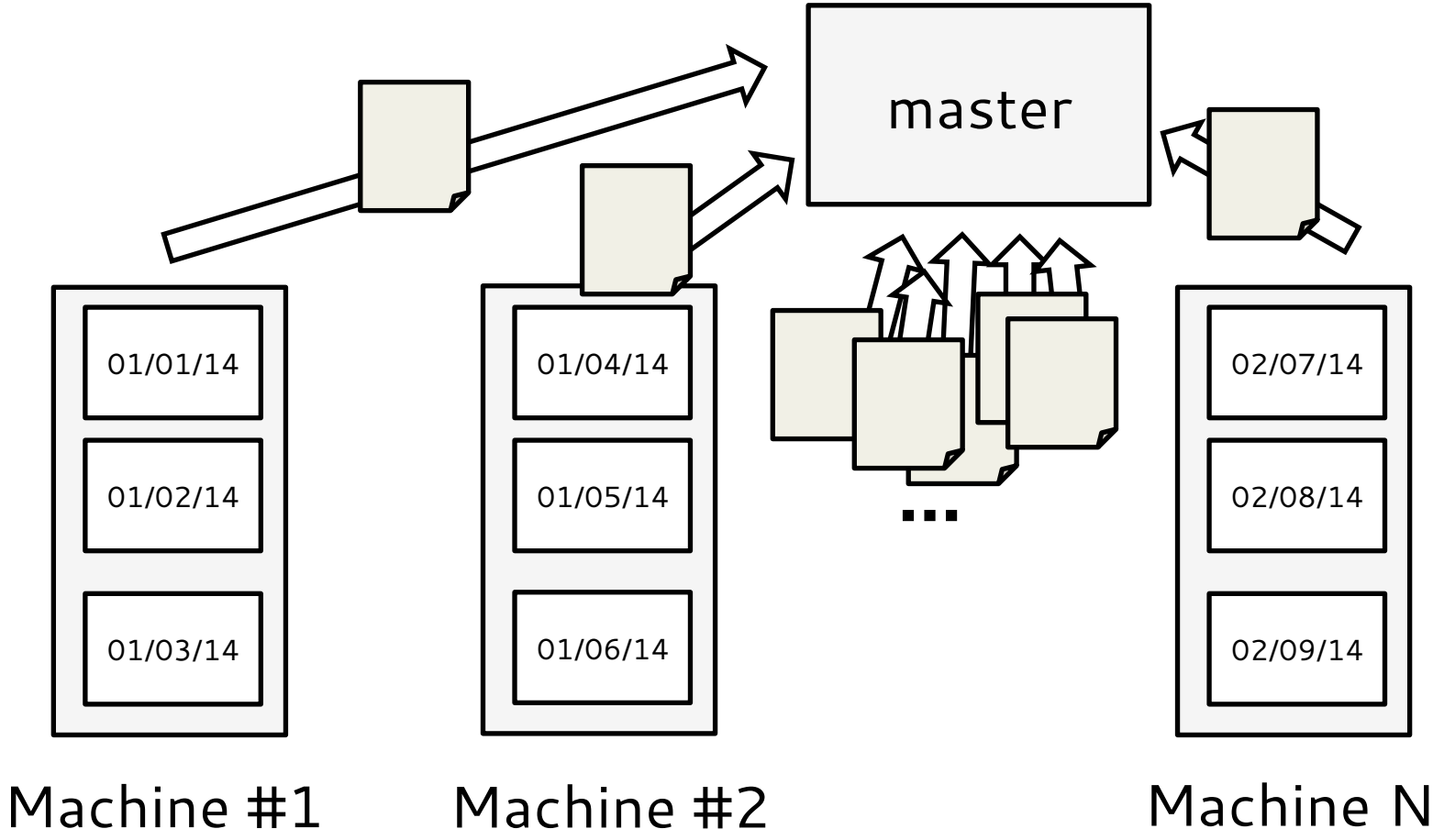
```
SELECT
count(distinct(ip))
FROM requests
WHERE time >= '2014-01-01
00:00'
AND time < '2014-02-01
00:00';
```

citusdata

What happens when you scale?



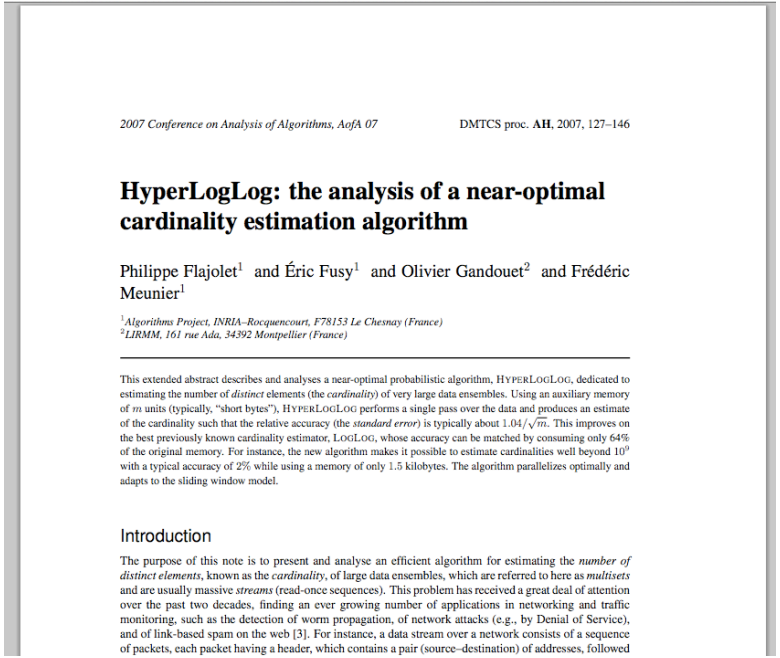
citusdata


master
01/01/14
01/02/14
01/03/14
01/04/14
01/05/14
01/06/14
...
02/07/14
02/08/14
02/09/14
Machine #1
Machine #2
Machine N

citusdata

Academia to the rescue!

- HyperLogLog (Flajolet et. al.) paper from 2007



2007 Conference on Analysis of Algorithms, AofA 07 DMTCS proc. **AH**, 2007, 127–146

HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm

Philippe Flajolet[1] and Éric Fusy[1] and Olivier Gandouet[2] and Frédéric Meunier[1]

[1] *Algorithms Project, INRIA–Rocquencourt, F78153 Le Chesnay (France)*
[2] *LIRMM, 161 rue Ada, 34392 Montpellier (France)*

This extended abstract describes and analyses a near-optimal probabilistic algorithm, HYPERLOGLOG, dedicated to estimating the number of *distinct* elements (the *cardinality*) of very large data ensembles. Using an auxiliary memory of m units (typically, "short bytes"), HYPERLOGLOG performs a single pass over the data and produces an estimate of the cardinality such that the relative accuracy (the *standard error*) is typically about $1.04/\sqrt{m}$. This improves on the best previously known cardinality estimator, LOGLOG, whose accuracy can be matched by consuming only 64% of the original memory. For instance, the new algorithm makes it possible to estimate cardinalities well beyond 10^9 with a typical accuracy of 2% while using a memory of only 1.5 kilobytes. The algorithm parallelizes optimally and adapts to the sliding window model.

Introduction

The purpose of this note is to present and analyse an efficient algorithm for estimating the *number of distinct elements*, known as the *cardinality*, of large data ensembles, which are referred to here as *multisets* and are usually massive *streams* (read-once sequences). This problem has received a great deal of attention over the past two decades, finding an ever growing number of applications in networking and traffic monitoring, such as the detection of worm propagation, of network attacks (e.g., by Denial of Service), and of link-based spam on the web [3]. For instance, a data stream over a network consists of a sequence of packets, each packet having a header, which contains a pair (source–destination) of addresses, followed

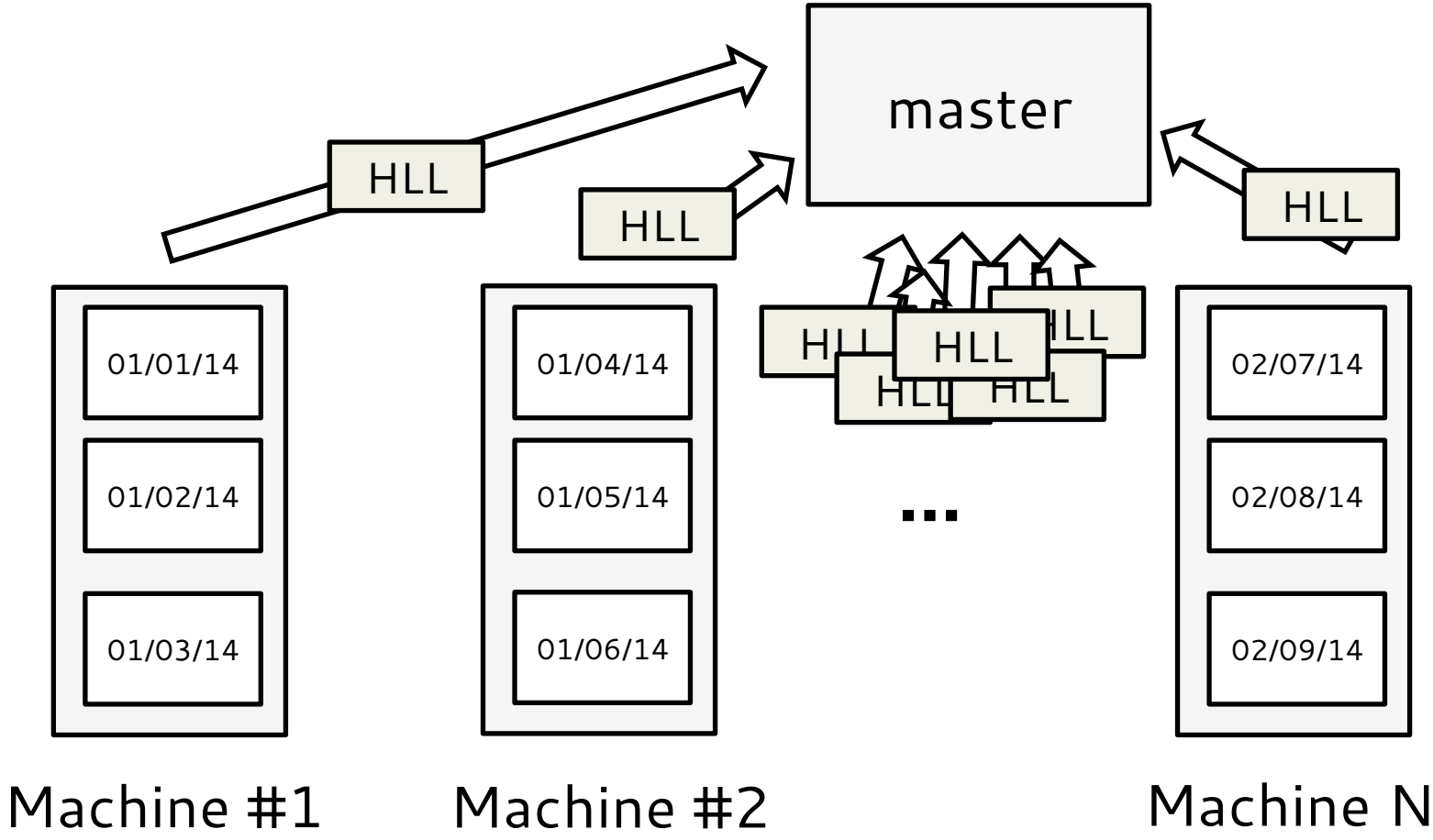
citusdata

HLL extension

- https://github.com/aggregateknowledge/postgresql-hll
- Enables approximate distinct counts
- Adds new type, aggregation functions, and estimation functions

HLL extension

- `hll_hash` – create HLL given a value
- `hll_union` – combine two HLLs
- `hll_cardinality` – provide an estimate of the number of distinct values in an HLL


master
HLL
HLL
HLL
HLL
HLL
HLL
HLL
01/01/14
01/02/14
01/03/14
01/04/14
01/05/14
01/06/14
...
02/07/14
02/08/14
02/09/14
Machine #1
Machine #2
Machine N

citusdata

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
    - HSTORE – semi-structured data in your DB
    - HLL – distinct counts using mathematical magic
    - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

citusdata

Id	Sz	Ln	Ht	…	…	…	…	…	…	…	…	…	…	…
1	4	3	4	…	…	…	…	…	…	…	…	…	…	…
2	4	11	3	…	…	…	…	…	…	…	…	…	…	…
3	1	4	2	…	…	…	…	…	…	…	…	…	…	…
4	8	4	12	…	…	…	…	…	…	…	…	…	…	…
…														
4…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
4…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
4…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

citusdata

```
SELECT
   id, AVG(price), MAX(price)
FROM
   items
WHERE
   quantity > 100 AND
   last_stock_date < ‘2013-10-01’
GROUP BY
   weight
```

Row-oriented store

Id	…	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	…	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	…	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	…	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	…	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	…	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	…	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Row-oriented store

Id	…	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	…	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	…	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	…	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	…	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	…	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	…	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Row-oriented store

Id	…	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	…	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	…	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	…	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	…	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	…	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	…	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Row-oriented store

Id	…	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	…	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	…	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	…	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	…	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	…	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	…	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Cost of row storage

- Read 700 columns instead of 5
- >39 GB of unnecessary I/O

Input Type	Estimated Input Rate	Cost to query performance
Memory	10 GB/s	**3.9 seconds**
SSD	600 MB/s	**>60 seconds**

```
SELECT
   id, AVG(price), MAX(price)
FROM
   items
WHERE
   quantity > 100 AND
   last_stock_date < ‘2013-10-01’
GROUP BY
   weight
```

Column-oriented store

Id	sz	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	4	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	3	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	2	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	4	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	19	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	2	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Column-oriented store

Id	sz	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	4	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	3	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	2	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	4	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	19	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	2	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Column-oriented store

Id	sz	price	…	…	quan…	…	…	last_st…	…	…	…	…	…	weight
1	4	3.90	…	…	31	…	…	2013-…	…	…	…	…	…	0.6
2	3	13	…	…	70	…	…	2010-…	…	…	…	…	…	0.8
3	2	4.25	…	…	432	…	…	2013-…	…	…	…	…	…	1
4	4	4	…	…	45	…	…	2013-…	…	…	…	…	…	6
…														
4…	19	95	…	…	37	…	…	2013-…	…	…	…	…	…	0.6
4…	2	59	…	…	90	…	…	2012-…	…	…	…	…	…	1.5

citusdata

Benefits

- Less I/O if reading subset of columns
- Better compression:
  - Less disk usage
  - Less I/O

But how in PostgreSQL?

- Foreign Data Wrappers (FDW) allow you to easily connect to any external data source
- Exist already for Mongo, Redis, CouchDB, JSON, Oracle, MySQL, others

Announcing CSTORE

- Open source columnar store built by Citus Data
- Releases March 14th for PostgreSQL and CitusDB users

citusdata

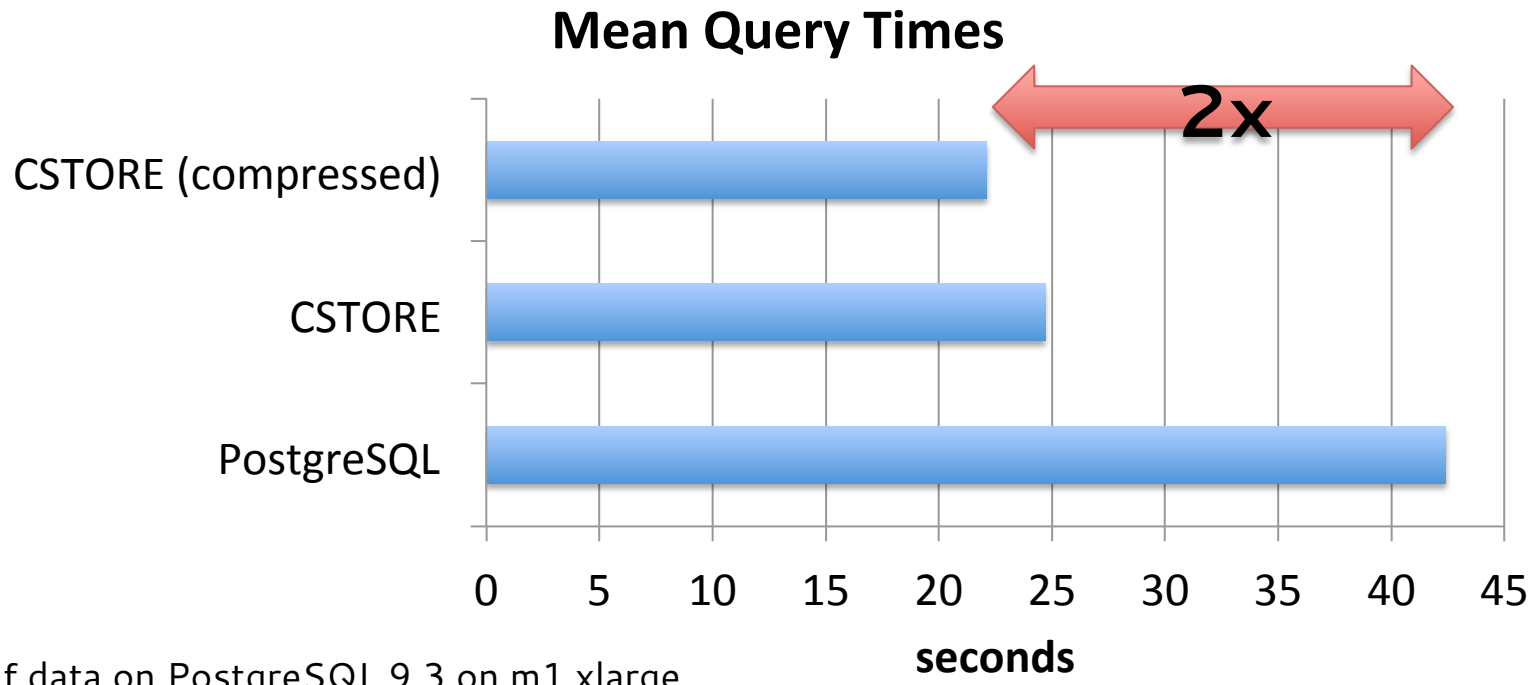
CSTORE features

- ORC inspired data layout benefits from years of learning with RCFile format
- Integrated with PostgreSQL FDW APIs:
  - Statistics collection for optimal query planning
  - Support for all PostgreSQL types and user defined types

CSTORE benchmarks

- TPC-H is the common benchmark
- Performed benchmarks with 4 GB of data on `m1.xlarge` instance
- Compared vanilla PostgreSQL, CSTORE, CSTORE with compression

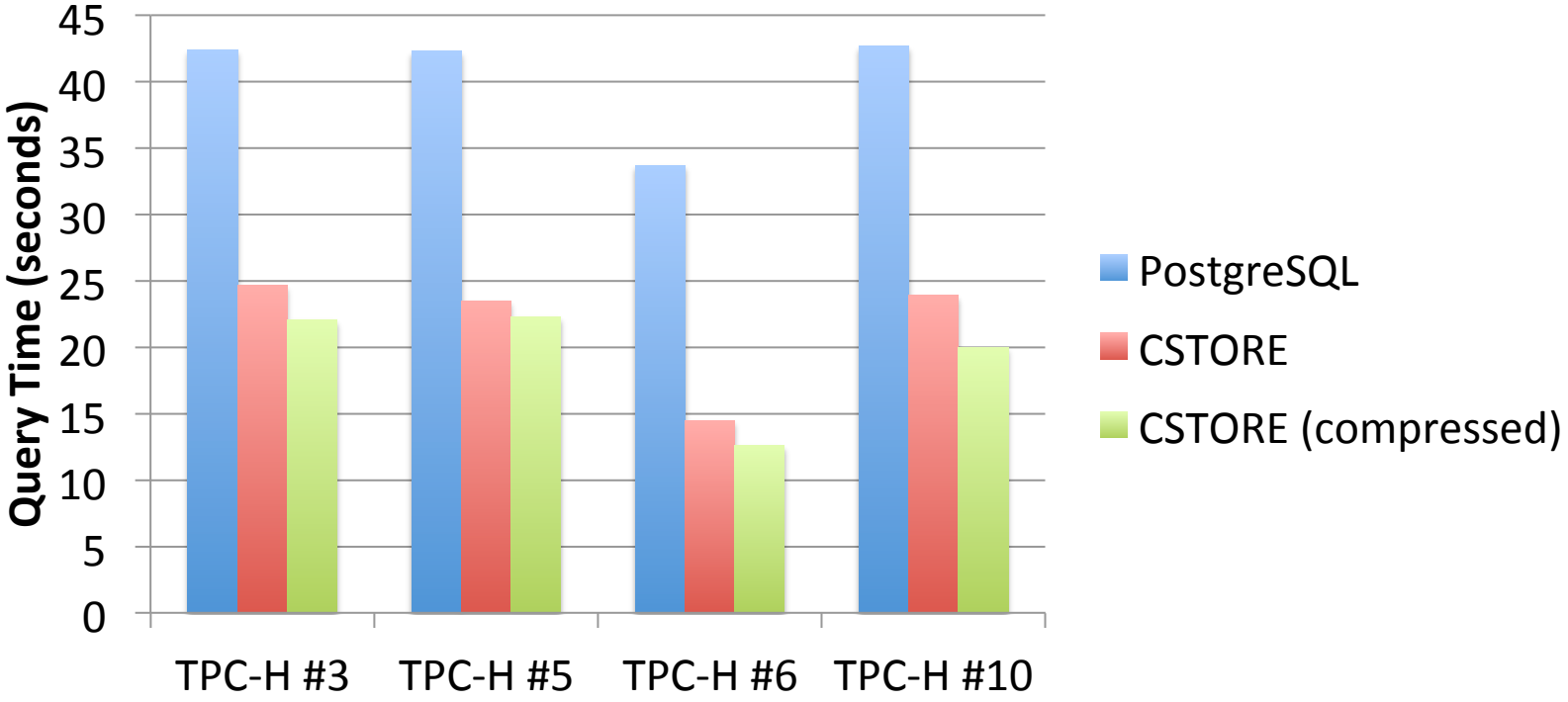
TPC-H Query 3



4GB of data on PostgreSQL 9.3 on m1.xlarge

citusdata

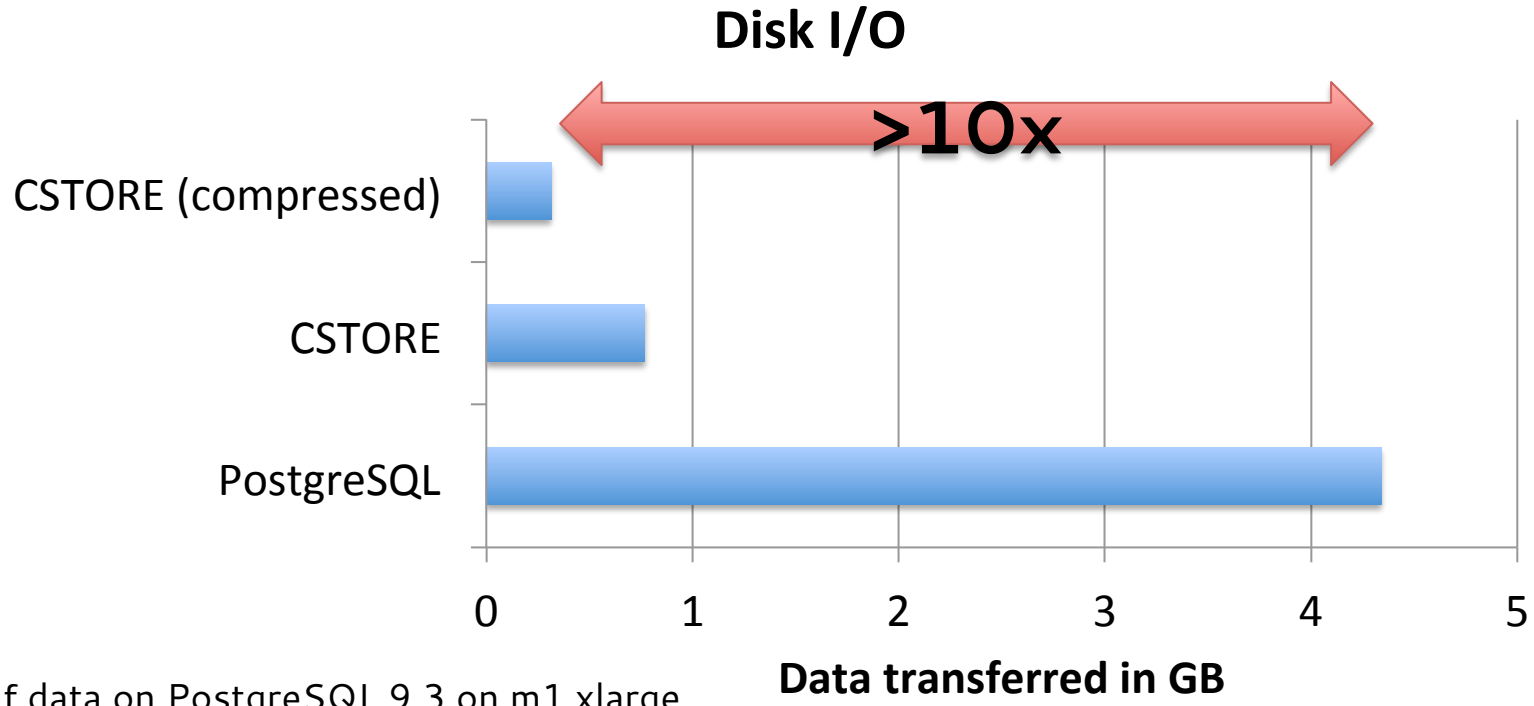
Other TPC-H Queries



4GB of data on PostgreSQL 9.3 on m1.xlarge

citusdata

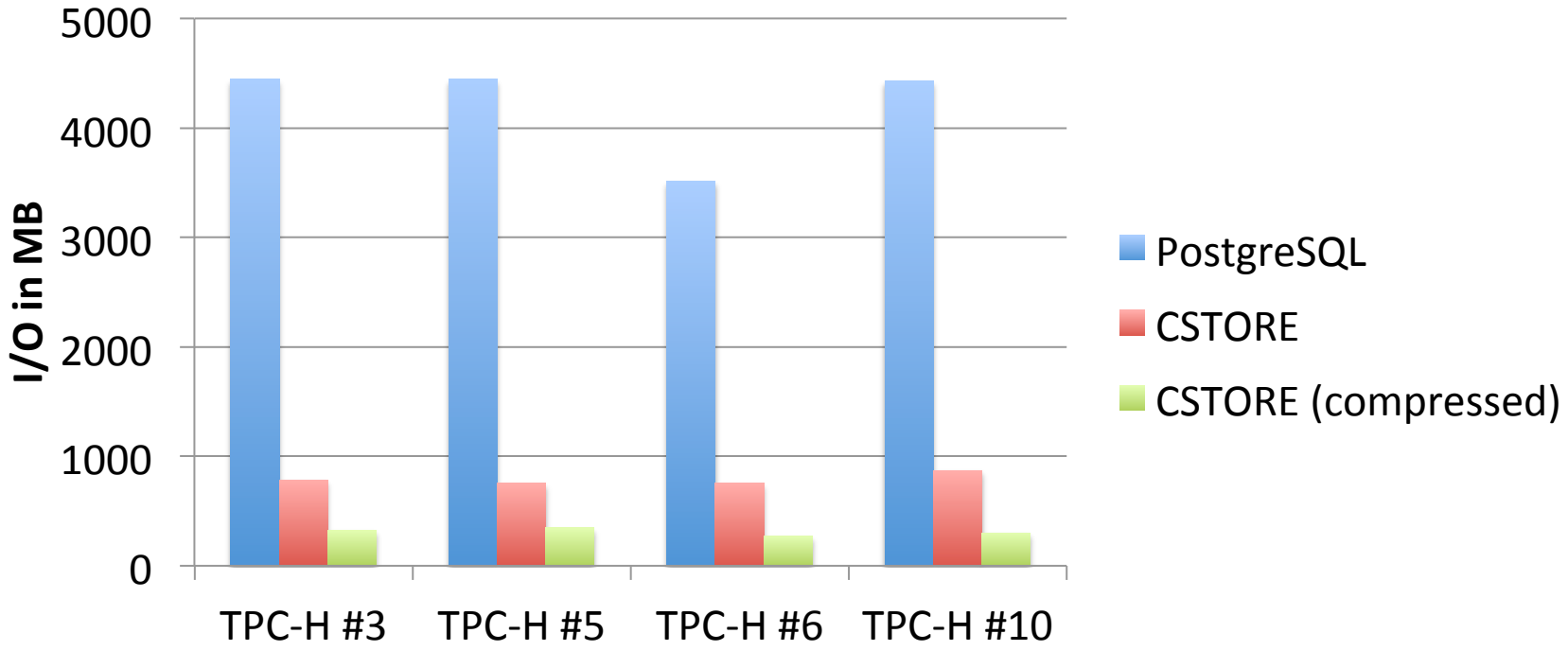
TPC-H Query 3



4GB of data on PostgreSQL 9.3 on m1.xlarge

citusdata

Other TPC-H Queries



4GB of data on PostgreSQL 9.3 on m1.xlarge

citusdata

1. What is a data platform?
2. Why PostgreSQL?
3. Extensions, Extensions, Extensions
    - HSTORE – semi-structured data in your DB
    - HLL – distinct counts using mathematical magic
    - CSTORE – a fast columnar store for PostgreSQL
4. How CitusDB lets you scale PostgreSQL

citusdata

Data Platform

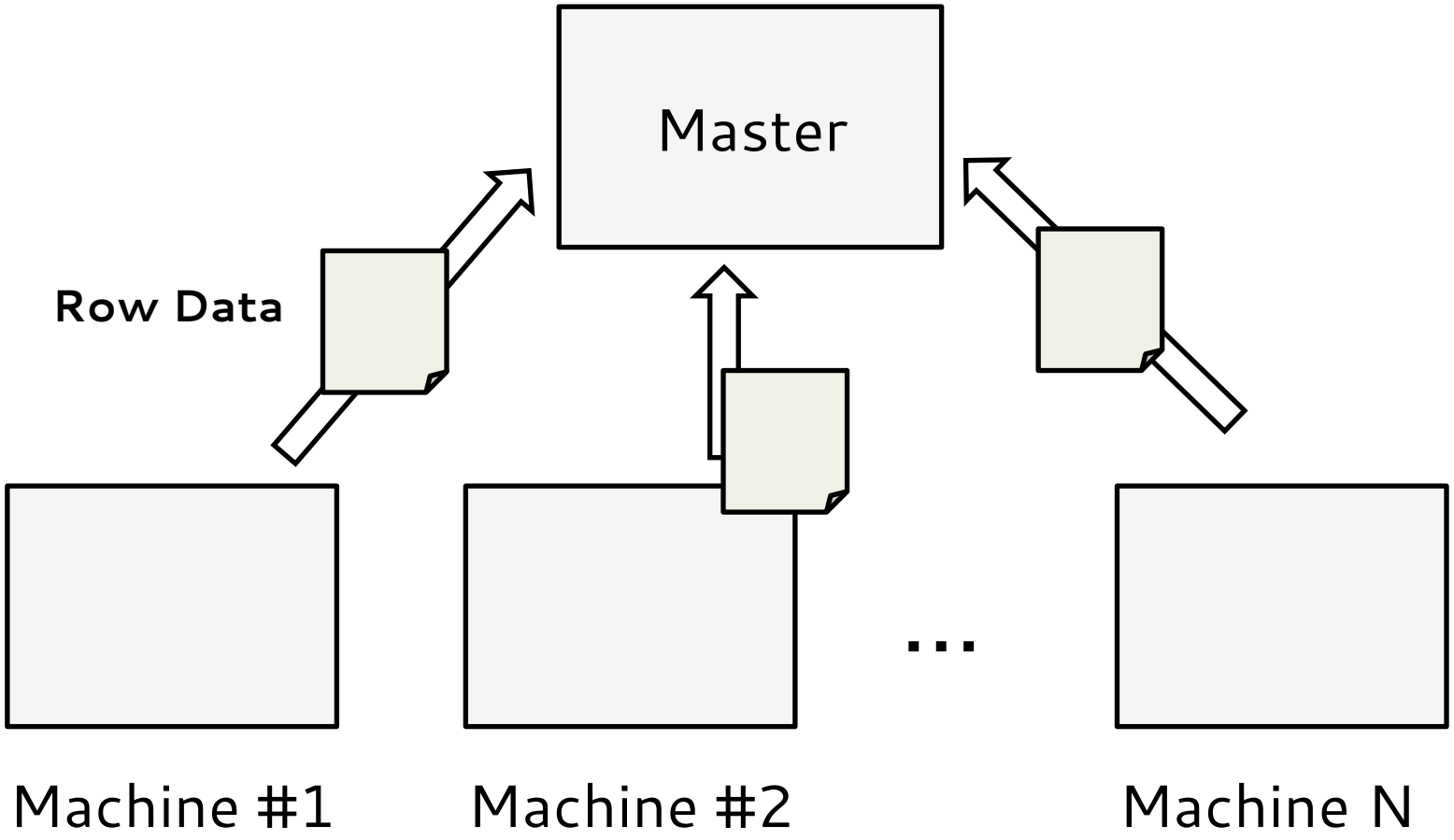
- Store and query ALL your data
- **Scalable**
- Cost effective
- Extensible

citusdata

Scaling Performance

- Minimize network I/O through advanced query parser integration
- Push compute to nodes

```
SELECT
   avg(price), max(price)
FROM
   items
WHERE
   quantity > 10
```

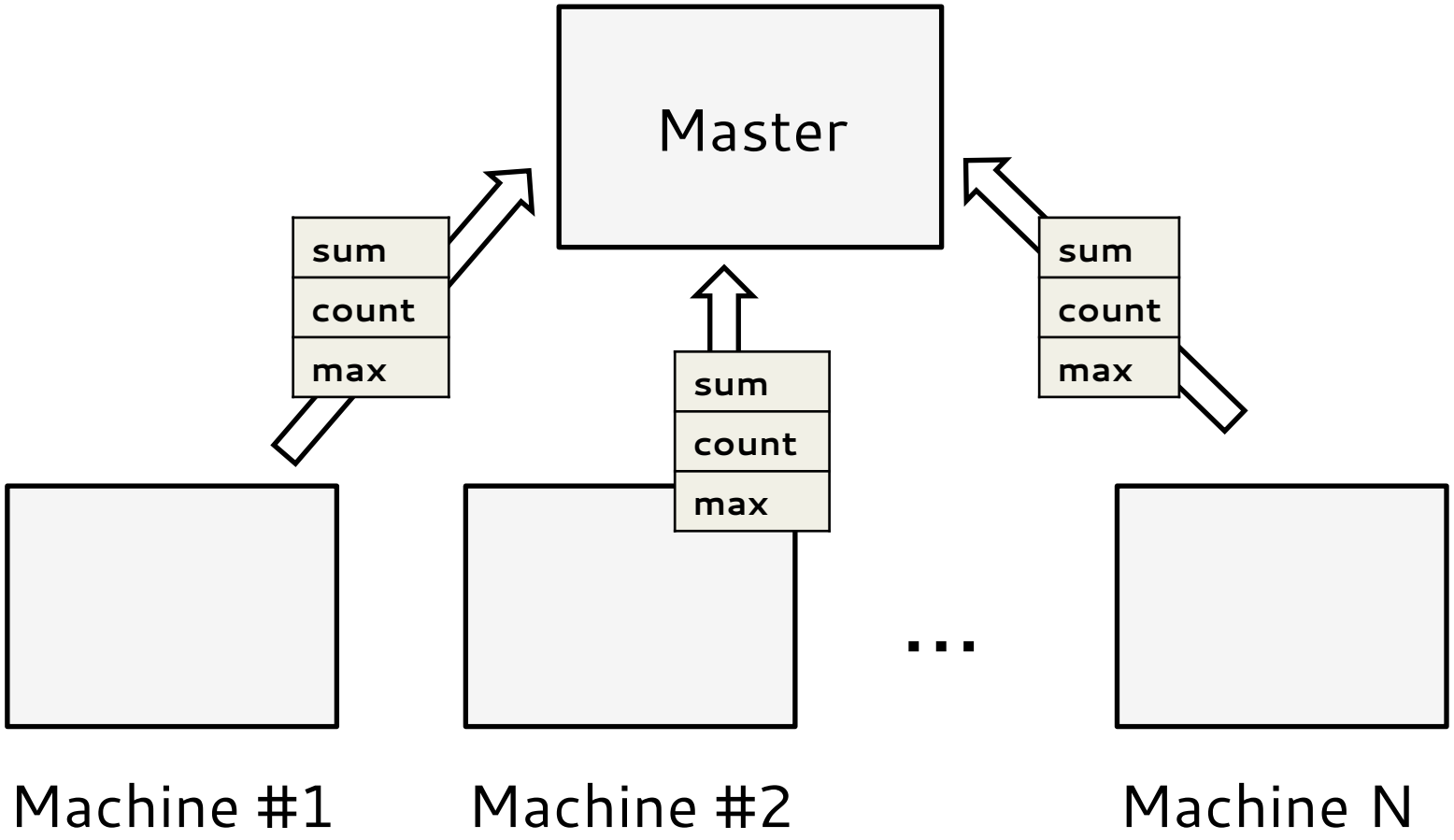


citusdata

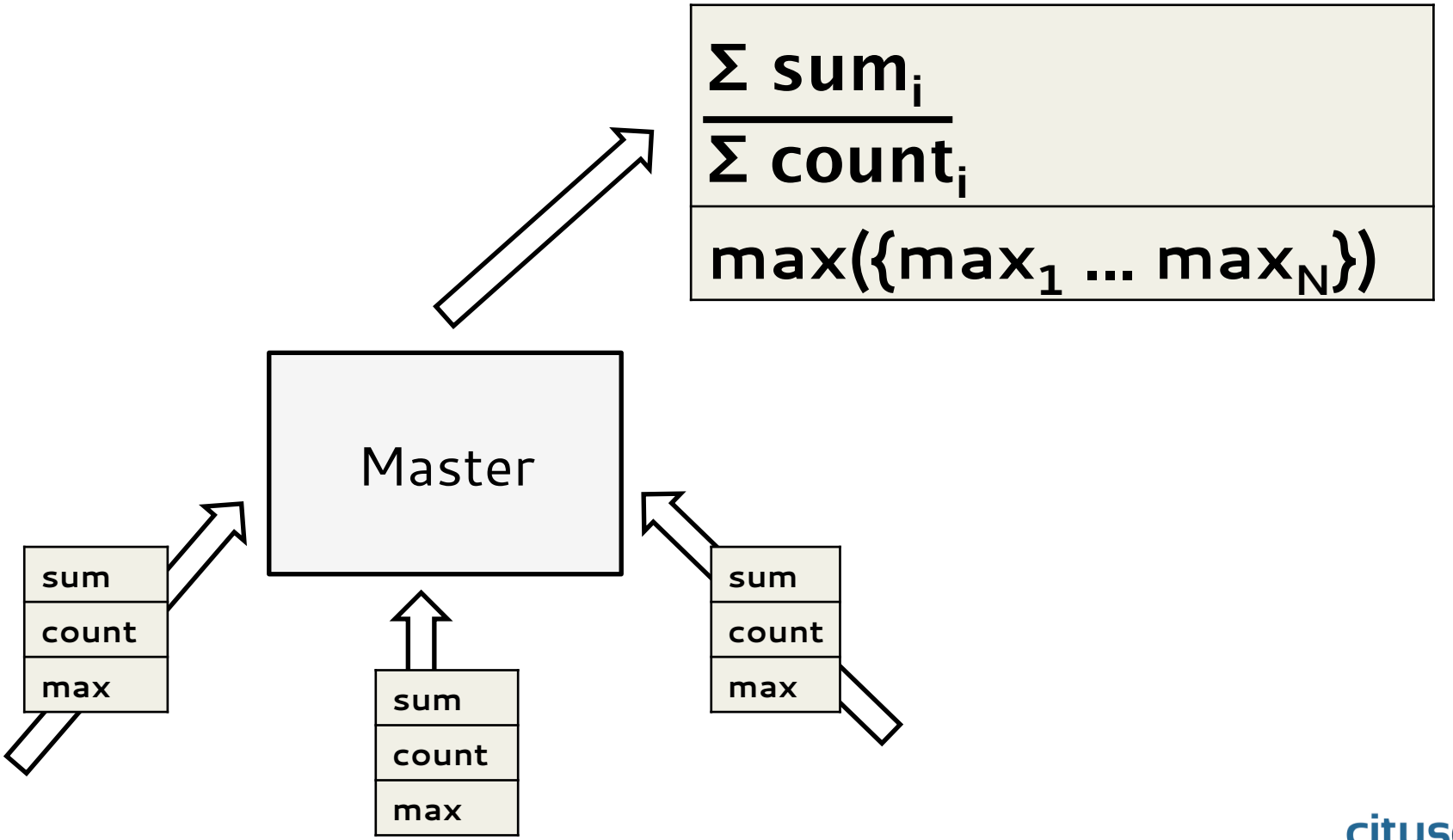
```
SELECT
   avg(price), max(price)
FROM
   items
WHERE
   quantity > 10
```

```
SELECT
   sum(price), count(*),
   max(price)
FROM
   items
WHERE
   quantity > 10
```

citusdata


Master
sum
count
max
sum
count
max
sum
count
max
...
Machine #1
Machine #2
Machine N

citusdata


$\frac{\Sigma\ sum_i}{\Sigma\ count_i}$
$max(\{max_1 \ldots max_N\})$
Master
sum
count
max
sum
count
max
sum
count
max

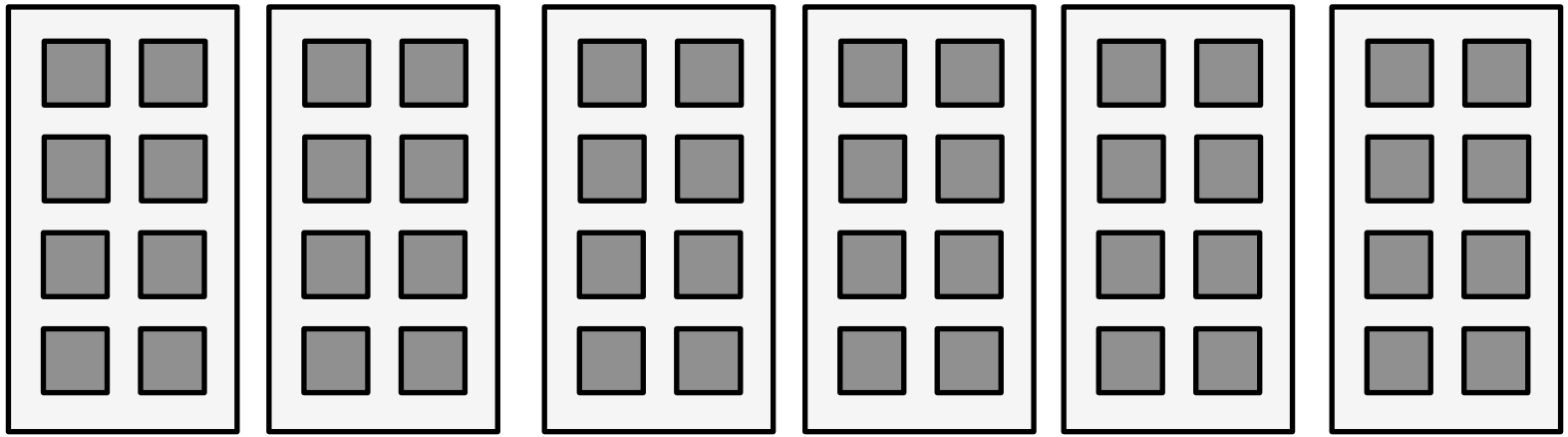
citusdata

Scaling Fault Tolerance

- Advanced query parser integration allows for partial query retries
- Block storage allows for improved node failure handling

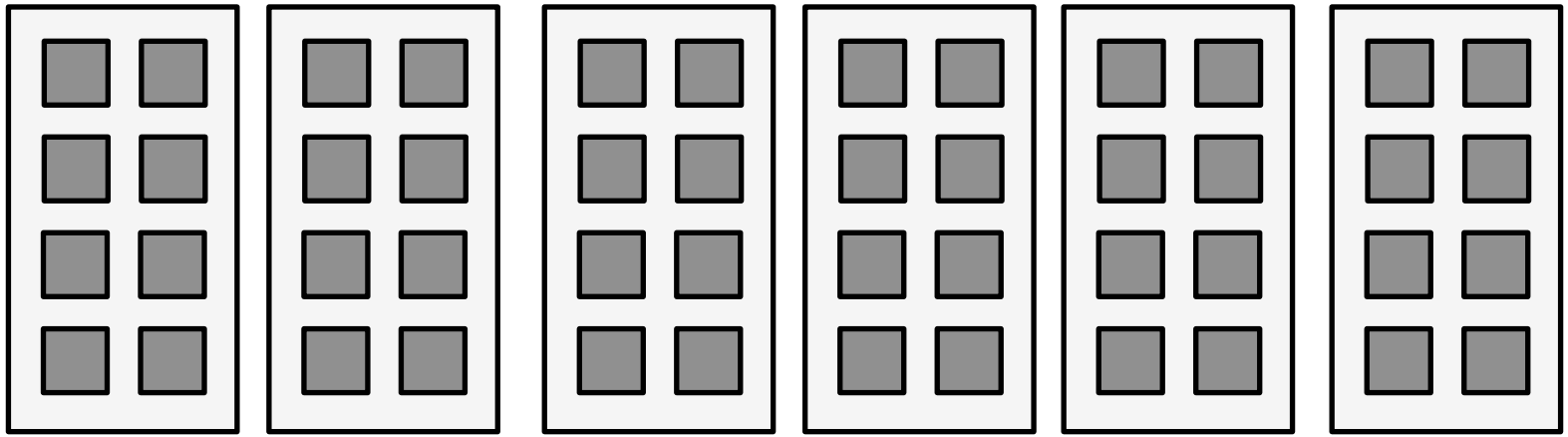
Fixed size block of data

Machine 1 ... Machine 6



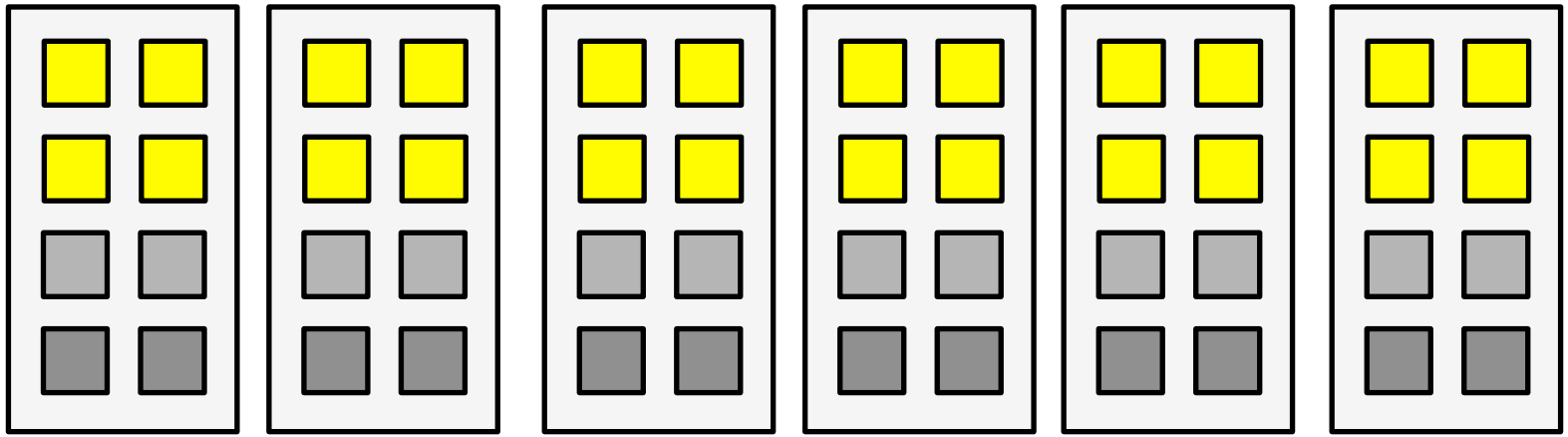
citusdata

```
SELECT
    avg(price), max(price)
FROM
    items
WHERE
    quantity > 10
```



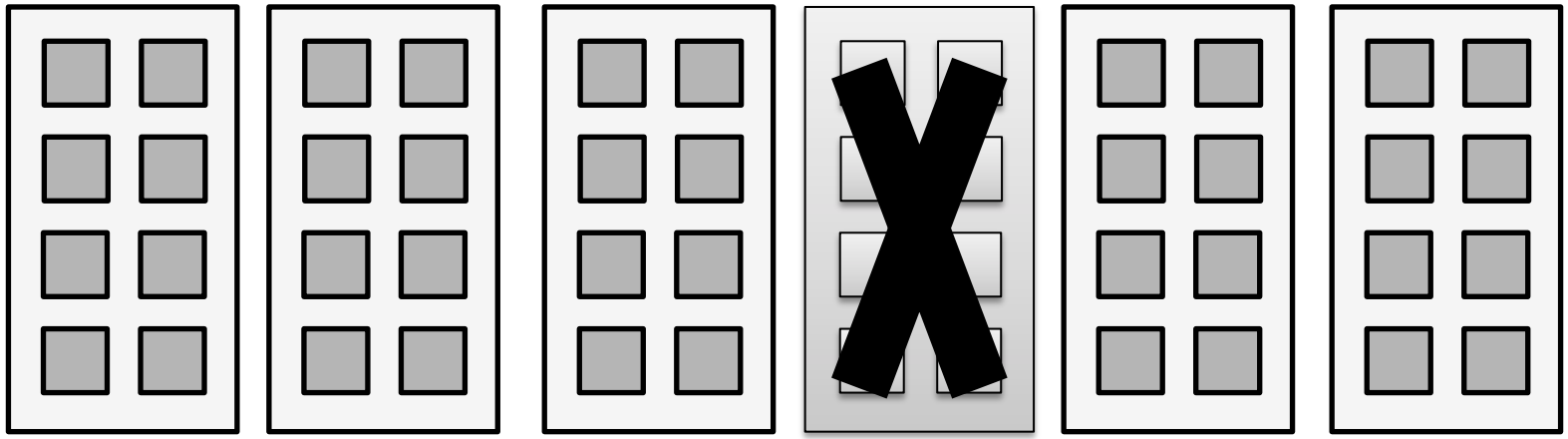
citusdata

```
SELECT
    avg(price), max(price)
FROM
    items
WHERE
    quantity > 10
```

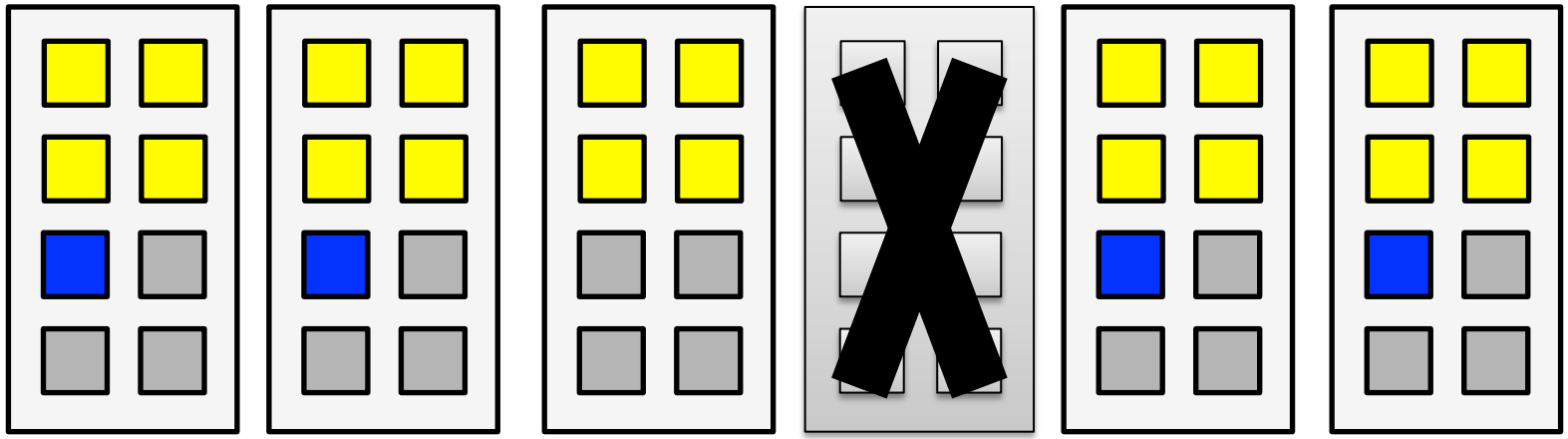


citusdata

```
SELECT
    avg(price), max(price)
FROM
    items
WHERE
    quantity > 10
```



```
SELECT
    avg(price), max(price)
FROM
    items
WHERE
    quantity > 10
```



Citus 3.0

- Support for large table joins
- Includes PostgreSQL 9.3 features (improved JSON support, etc.)
- Available Feb 23rd

Summary

- PostgreSQL makes a great extensible single node solution
- For it to be a data platform it needs to scale
- Citus makes PostgreSQL scale

citusdata

Acknowledgements

- PostgreSQL http://www.postgresql.org/
- Heap https://heapanalytics.com/
- HLL http://www.aggregateknowledge.com/
- Icons http://www.iconfinder.com/tmthymllr

citusdata

Scalable PostgreSQL as your data platform

Ben Redman – ben@citusdata.com