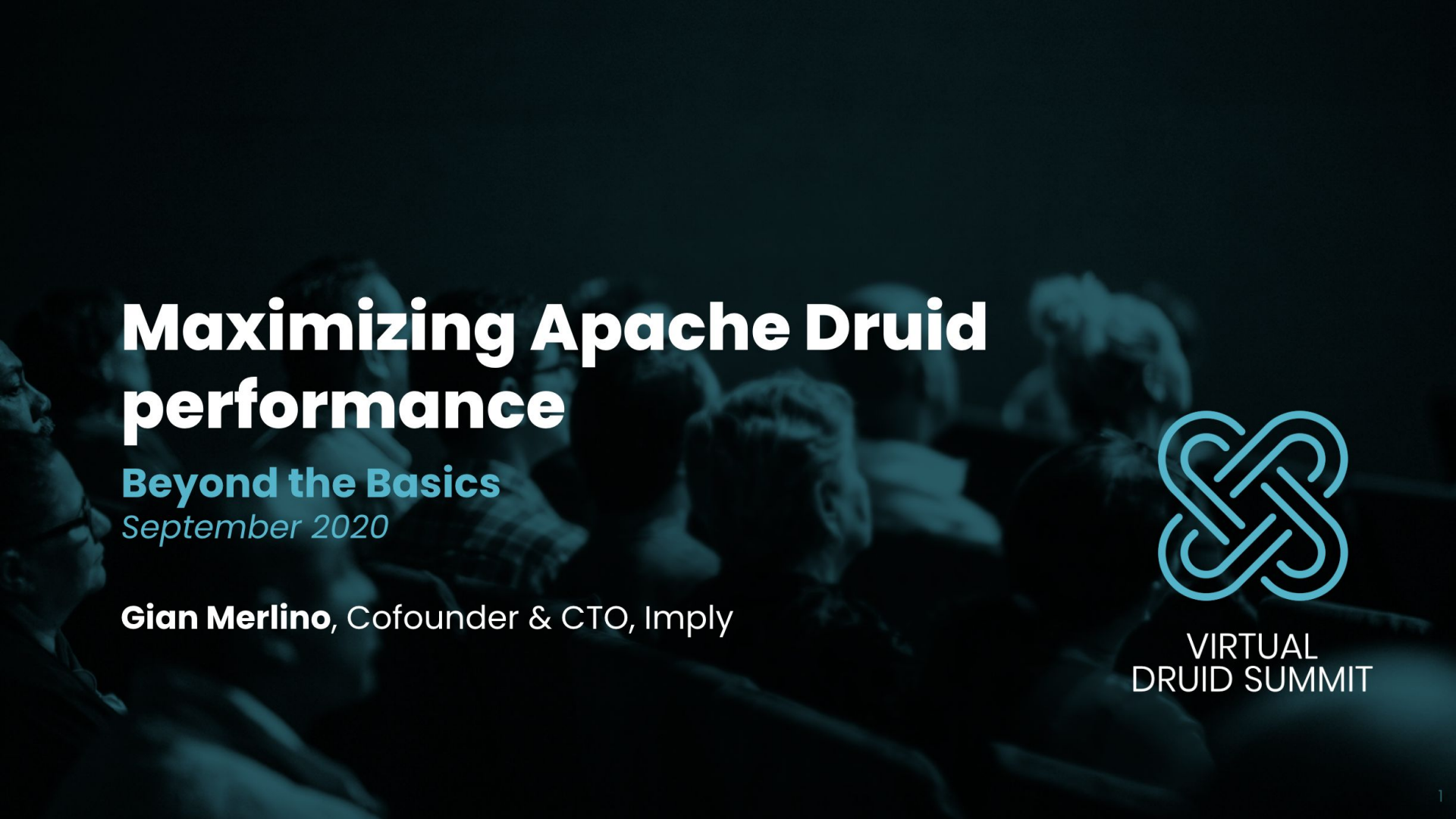


Maximizing Apache Druid performance

Beyond the Basics

September 2020

Gian Merlino, Cofounder & CTO, Imply

VIRTUAL
DRUID SUMMIT

Who am I?

Gian Merlino 

Committer & PMC chair at

Cofounder at imply (we're hiring!)

Druid in the wild

VIRTUAL
DRUID SUMMIT

Druid powers analytic applications

VIRTUAL DRUID SUMMIT

looker

Metabase

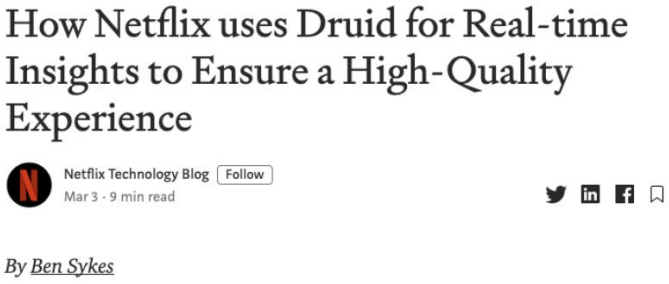
SQL

Superset

druid

Druid in the wild

VIRTUAL
DRUID SUMMIT


How Netflix uses Druid for Real-time Insights to Ensure a High-Quality Experience

Netflix Technology Blog Follow
Mar 3 · 9 min read

By Ben Sykes


Interactive Analytics at MoPub: Querying Terabytes of Data in Seconds

By Aaron Rolett and Shravana Krishnamurthy
Wednesday, 3 July 2019


How Druid enables analytics at Airbnb

Realtime and batch analytics at Airbnb and the role Druid plays in our analytics system architecture

Pala Follow
Nov 13, 2018 · 10 min read

By Pala Muthiah and Jinyang Li

100+ billion rows/day

1+ trillion rows, 1+ year retained

100s of servers

sub-second to few seconds query latency

mix of streaming and batch ingest

Hot vs. cold

VIRTUAL DRUID SUMMIT

🔥

- ⏱️ 0.1–3s query
- 🚰 fresh data
- 🏋️ high concurrency
- 🚴 highly interactive (OLAP)

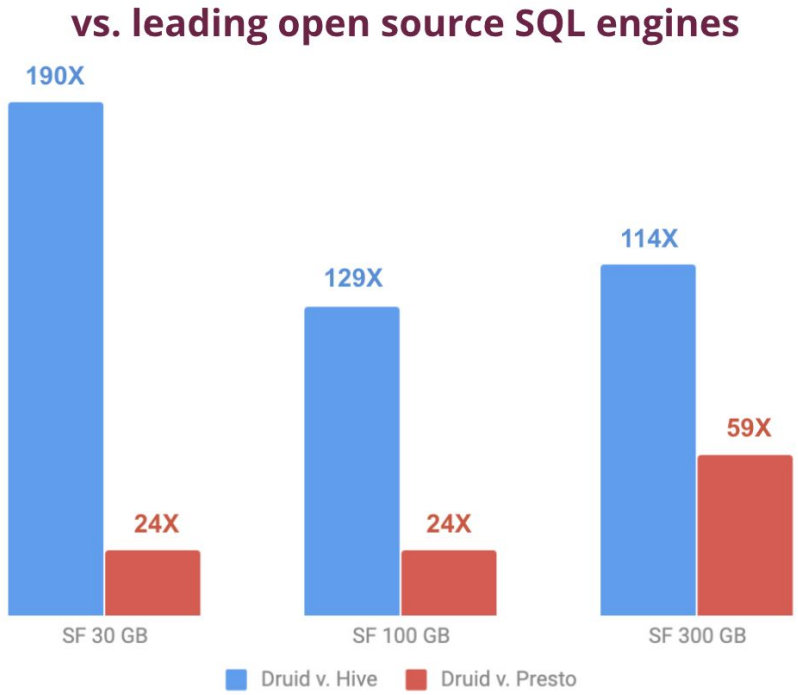
❄️

- ⏱️ slow queries are ok
- 🚰 less fresh data is ok
- 🏋️ low concurrency
- 🚴 reporting / planning


Designed for performance
VIRTUAL
DRUID SUMMIT

Druid is designed for performance

VIRTUAL
DRUID SUMMIT

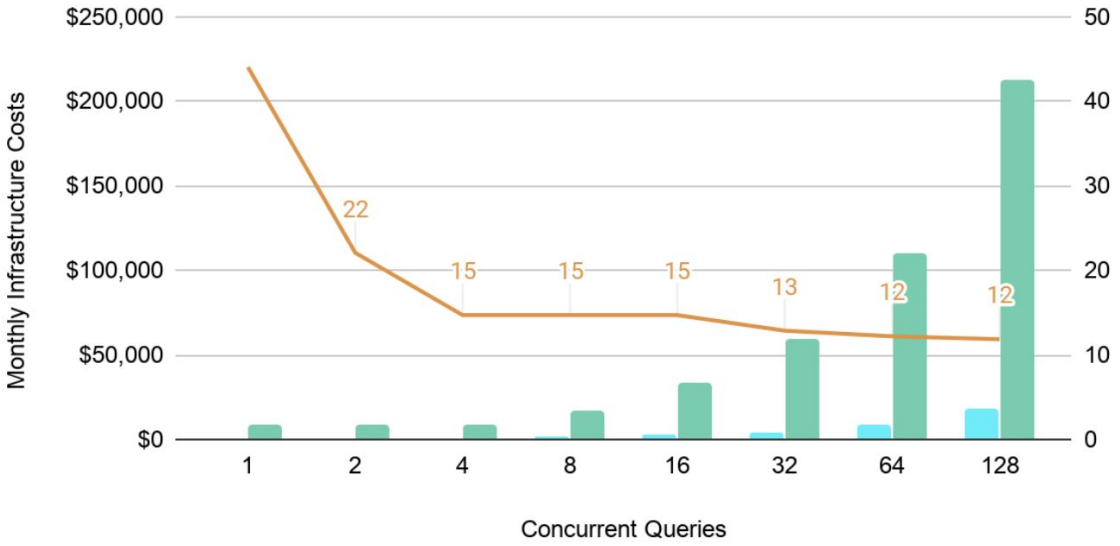


Data sourced from: *Correia, José & Costa, Carlos & Santos, Maribel. (2019). Challenging SQL-on-Hadoop Performance with Apache Druid.*

Druid is designed for performance

VIRTUAL DRUID SUMMIT

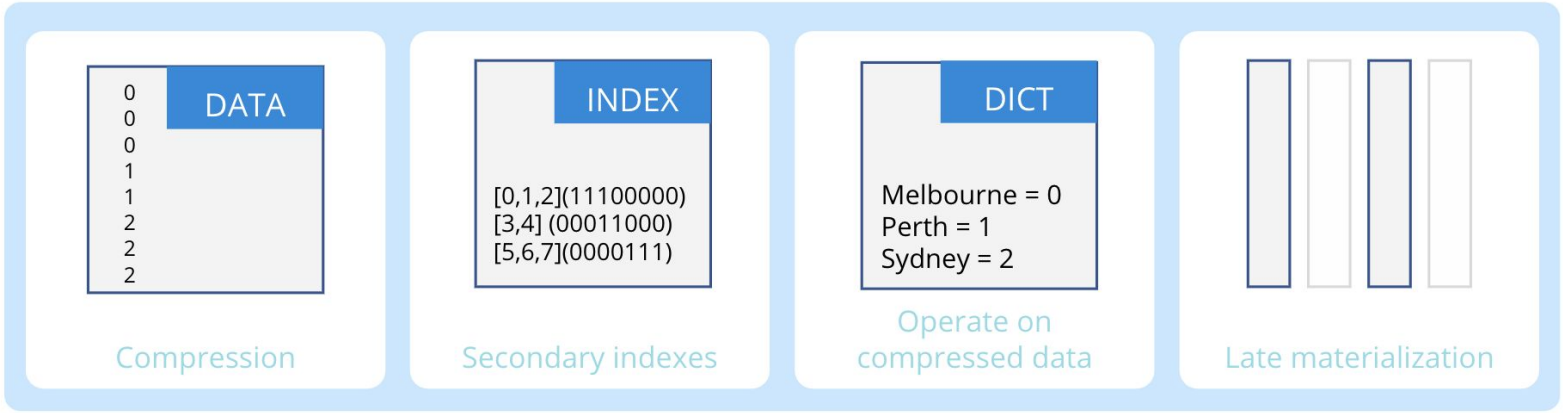
vs. a leading cloud-based data warehouse



Data sourced from Imply benchmarks.

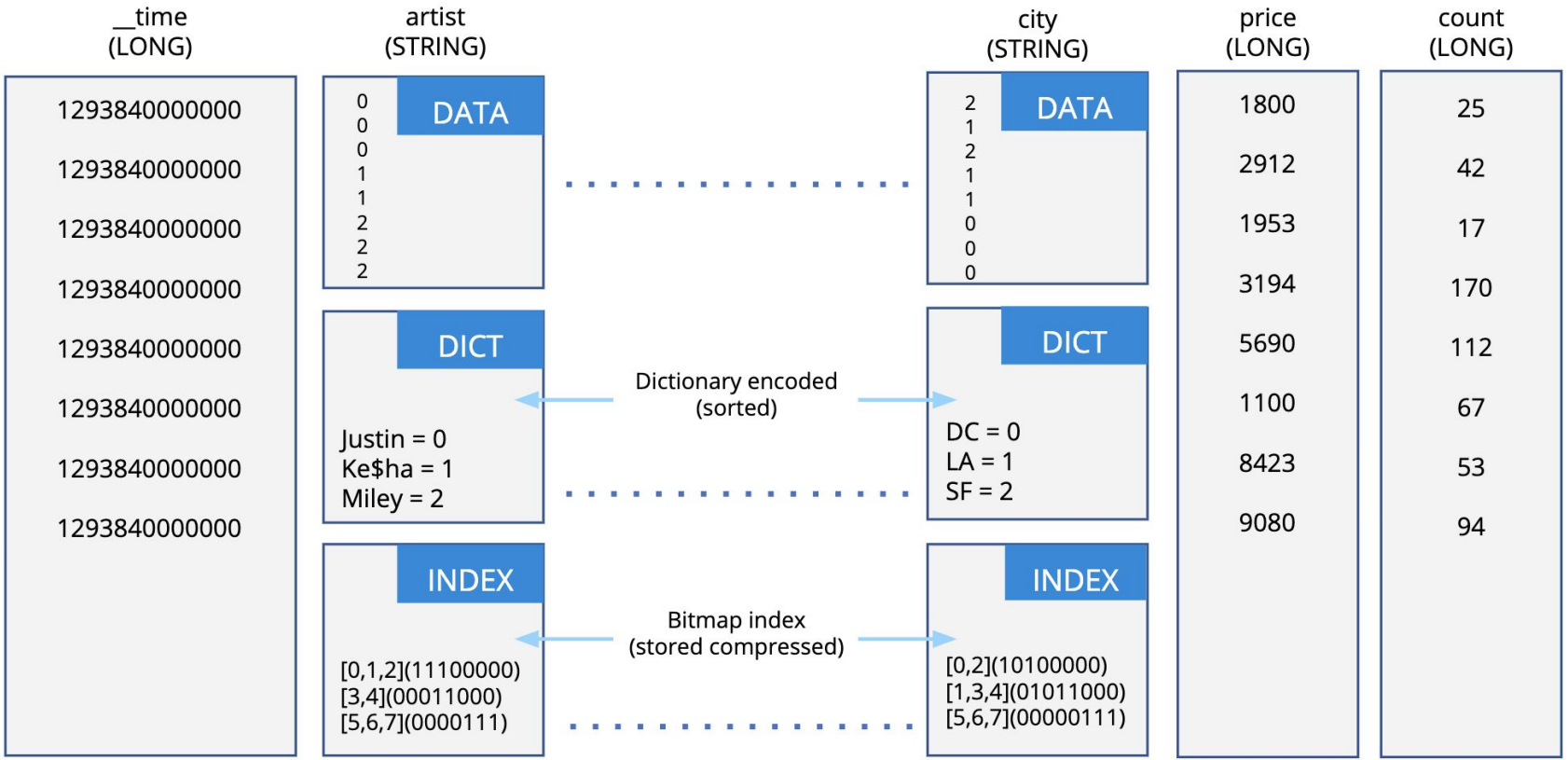
Integrated engine and data format

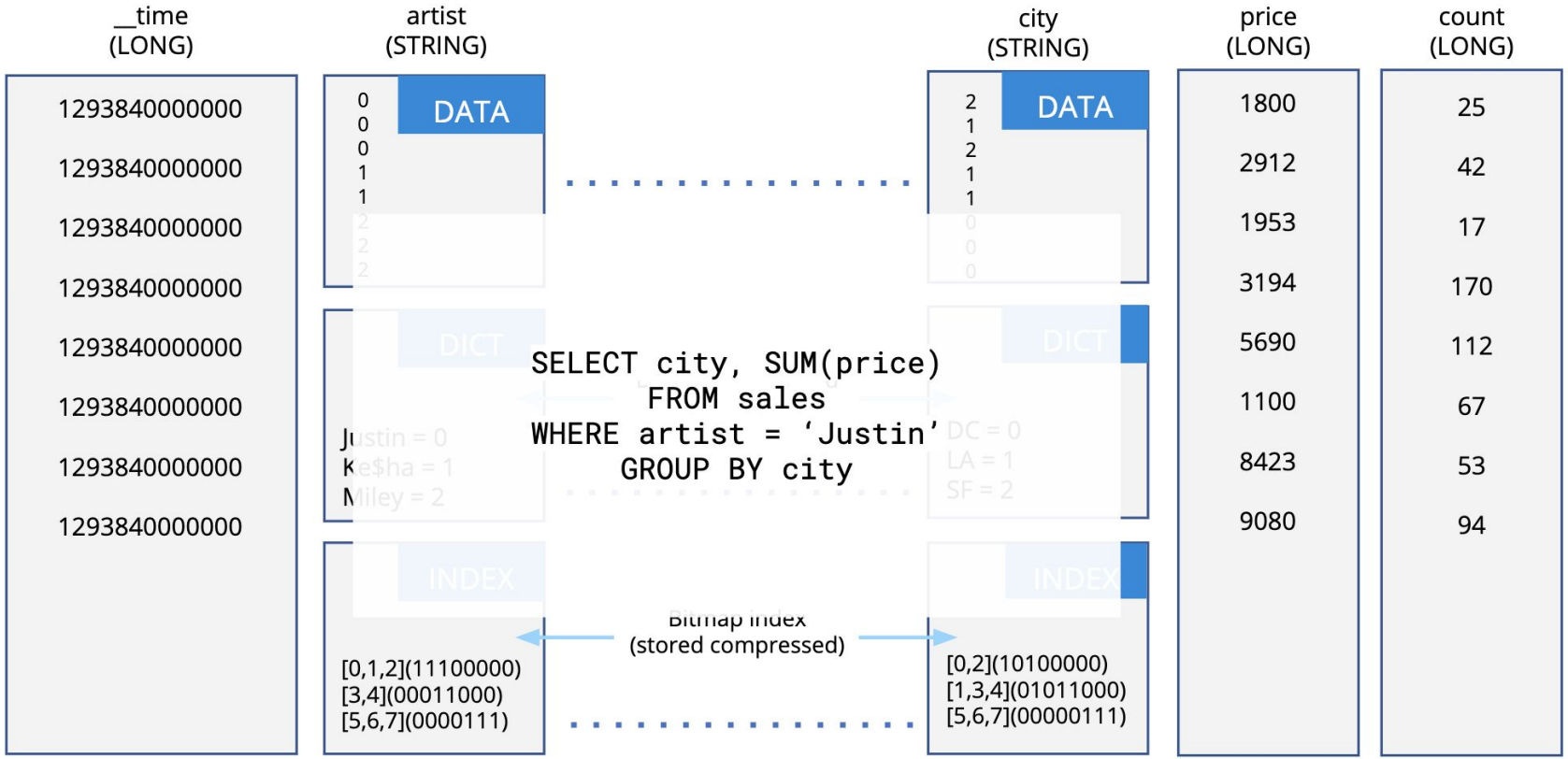
VIRTUAL DRUID SUMMIT



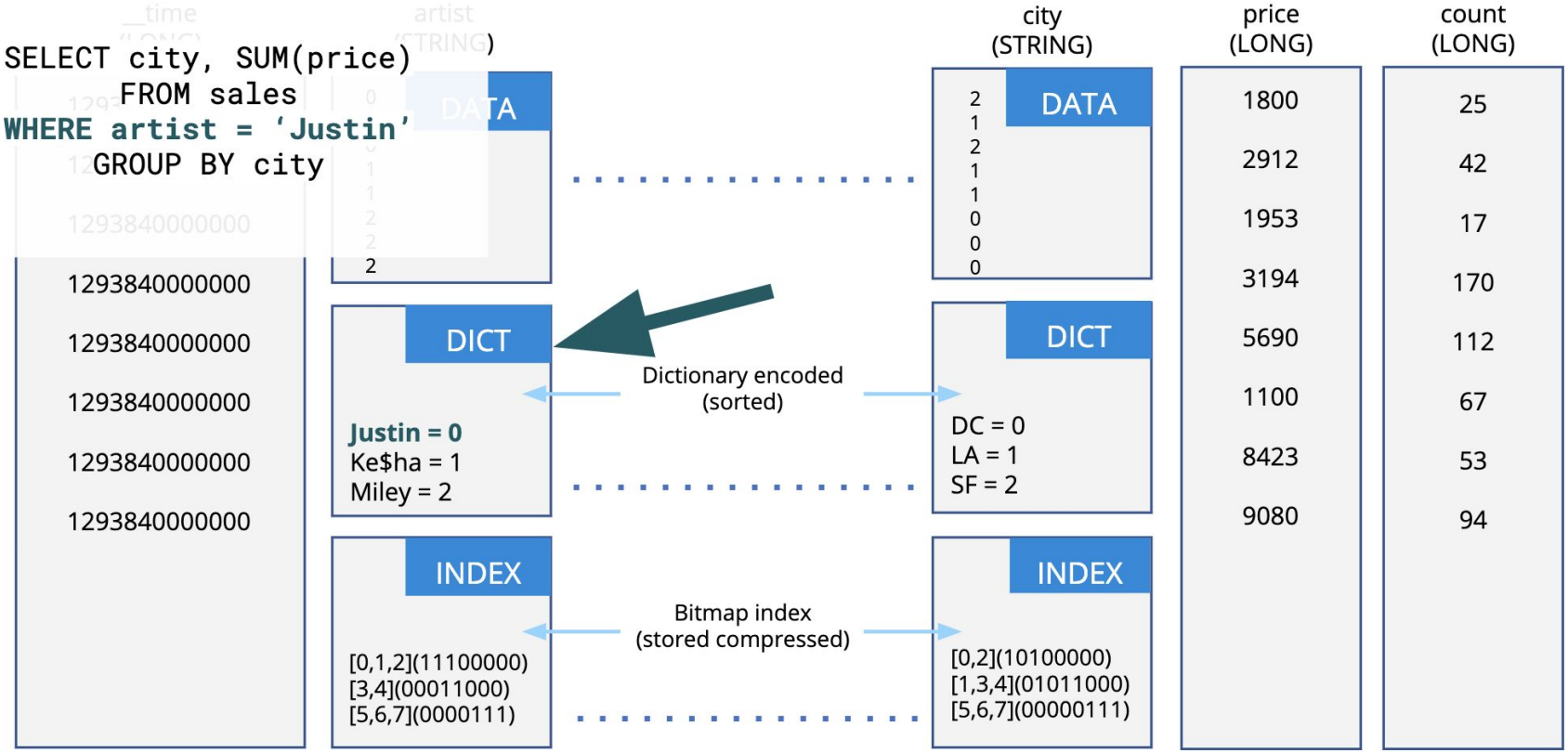
Query execution pipeline

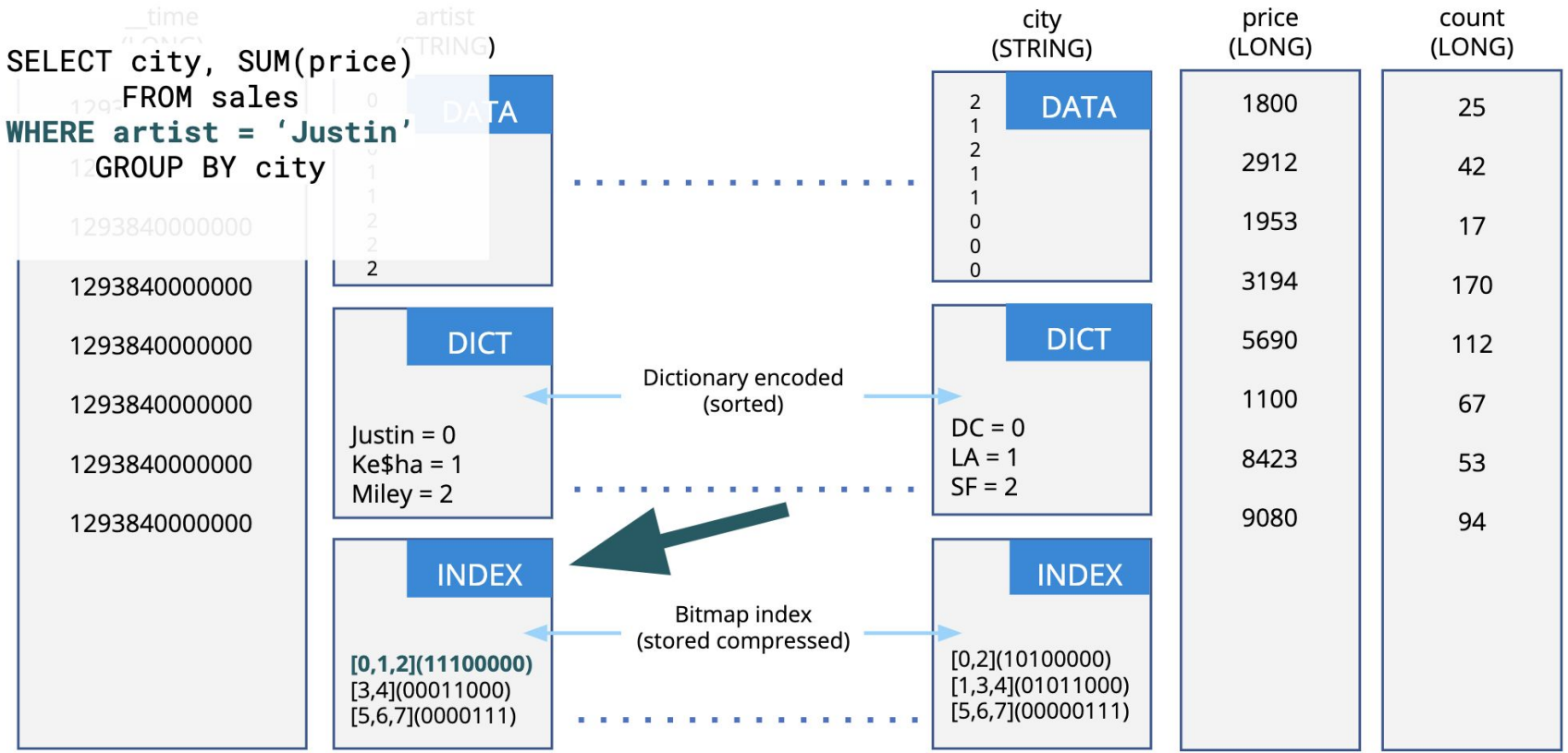
VIRTUAL DRUID SUMMIT

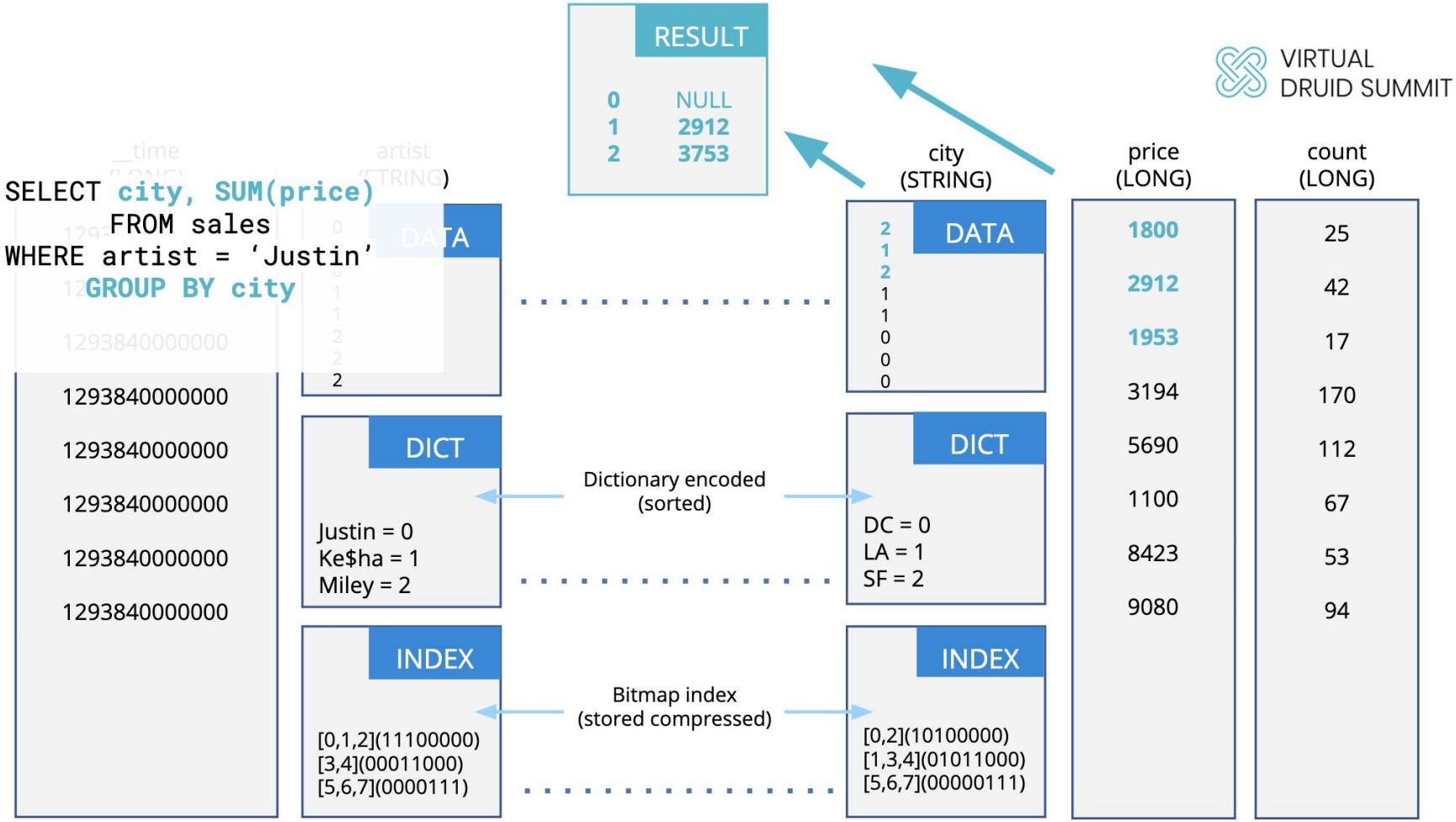



__time (LONG)
artist (STRING)
city (STRING)
price (LONG)
count (LONG)
1293840000000
1293840000000
1293840000000
1293840000000
1293840000000
1293840000000
1293840000000
1293840000000
DATA
0 0 0 1 1 2 2 2
DICT
Justin = 0
Ke$ha = 1
Miley = 2
INDEX
[0,1,2](11100000)
[3,4](00011000)
[5,6,7](0000111)
DATA
2 1 2 1 1 0 0 0
DICT
DC = 0
LA = 1
SF = 2
INDEX
[0,2](10100000)
[1,3,4](01011000)
[5,6,7](00000111)
1800
2912
1953
3194
5690
1100
8423
9080
25
42
17
170
112
67
53
94
SELECT city, SUM(price)
FROM sales
WHERE artist = 'Justin'
GROUP BY city
Bitmap index
(stored compressed)

```
SELECT city, SUM(price)
      FROM sales
WHERE artist = 'Justin'
     GROUP BY city
```




SELECT city, SUM(price)
FROM sales
WHERE artist = 'Justin'
GROUP BY city
1293840000000
1293840000000
1293840000000
1293840000000
1293840000000
DATA
2
2
DICT
Justin = 0
Ke$ha = 1
Miley = 2
INDEX
[0,1,2](11100000)
[3,4](00011000)
[5,6,7](0000111)
Dictionary encoded
(sorted)
Bitmap index
(stored compressed)
city
(STRING)
DATA
2
1
2
1
1
0
0
0
DICT
DC = 0
LA = 1
SF = 2
INDEX
[0,2](10100000)
[1,3,4](01011000)
[5,6,7](00000111)
price
(LONG)
1800
2912
1953
3194
5690
1100
8423
9080
count
(LONG)
25
42
17
170
112
67
53
94


SELECT city, SUM(price)
FROM sales
WHERE artist = 'Justin'
GROUP BY city
RESULT
0 NULL
1 2912
2 3753
VIRTUAL DRUID SUMMIT
__time (LONG)
1293840000000
1293840000000
1293840000000
1293840000000
1293840000000
artist (STRING)
DATA
0 1 1 2 2 2
DICT
Justin = 0
Ke$ha = 1
Miley = 2
INDEX
[0,1,2](11100000)
[3,4](00011000)
[5,6,7](0000111)
Dictionary encoded (sorted)
Bitmap index (stored compressed)
city (STRING)
DATA
2 1 2 1 1 0 0 0
DICT
DC = 0
LA = 1
SF = 2
INDEX
[0,2](10100000)
[1,3,4](01011000)
[5,6,7](00000111)
price (LONG)
1800
2912
1953
3194
5690
1100
8423
9080
count (LONG)
25
42
17
170
112
67
53
94

```
SELECT city, SUM(price)
      FROM sales
WHERE artist = 'Justin'
   GROUP BY city
```

RESULT

0	NULL
LA	2912
SF	3753

__time (LONG)

1293840000000
1293840000000
1293840000000
1293840000000
1293840000000

artist (STRING)

DATA
0
1
1
2
2
2

DICT
Justin = 0
Ke$ha = 1
Miley = 2

INDEX
[0,1,2](11100000)
[3,4](00011000)
[5,6,7](0000111)

Dictionary encoded (sorted)

Bitmap index (stored compressed)

city (STRING)

DATA
2
1
2
1
1
0
0
0

DICT
DC = 0
LA = 1
SF = 2

INDEX
[0,2](10100000)
[1,3,4](01011000)
[5,6,7](00000111)

price (LONG)

1800
2912
1953
3194
5690
1100
8423
9080

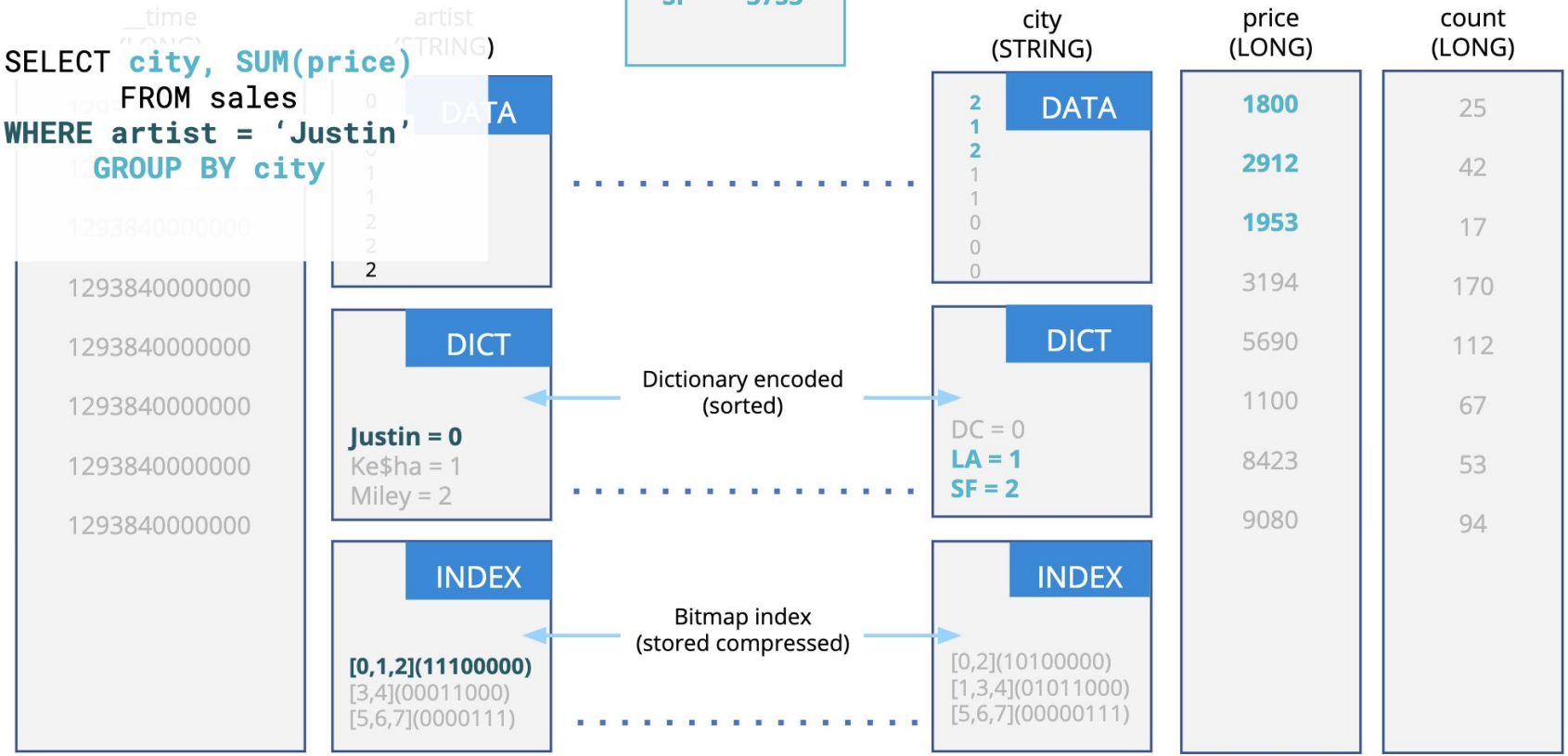
count (LONG)

25
42
17
170
112
67
53
94

```
SELECT city, SUM(price)
      FROM sales
WHERE artist = 'Justin'
    GROUP BY city
```

RESULT

LA	2912
SF	3753




Beyond the basics
VIRTUAL
DRUID SUMMIT

Getting my types right

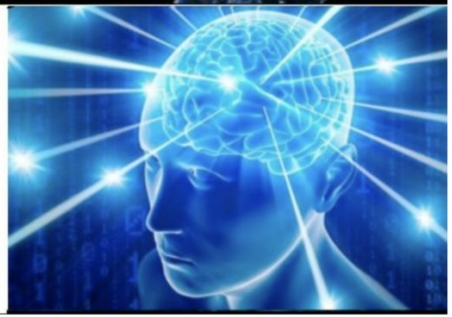
Good segment sizes

Denormalization

Using approximations when appropriate

Mastering data layout

Parachuting computation behind enemy lines



Getting my types right

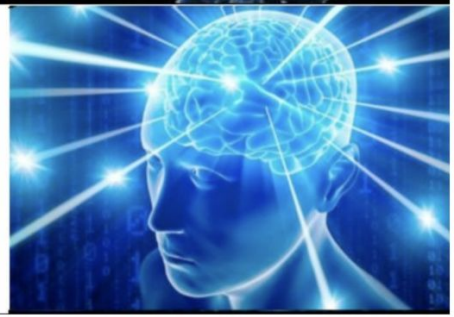
Good segment sizes

Denormalization

Using approximations when appropriate

Mastering data layout

Parachuting computation behind enemy lines

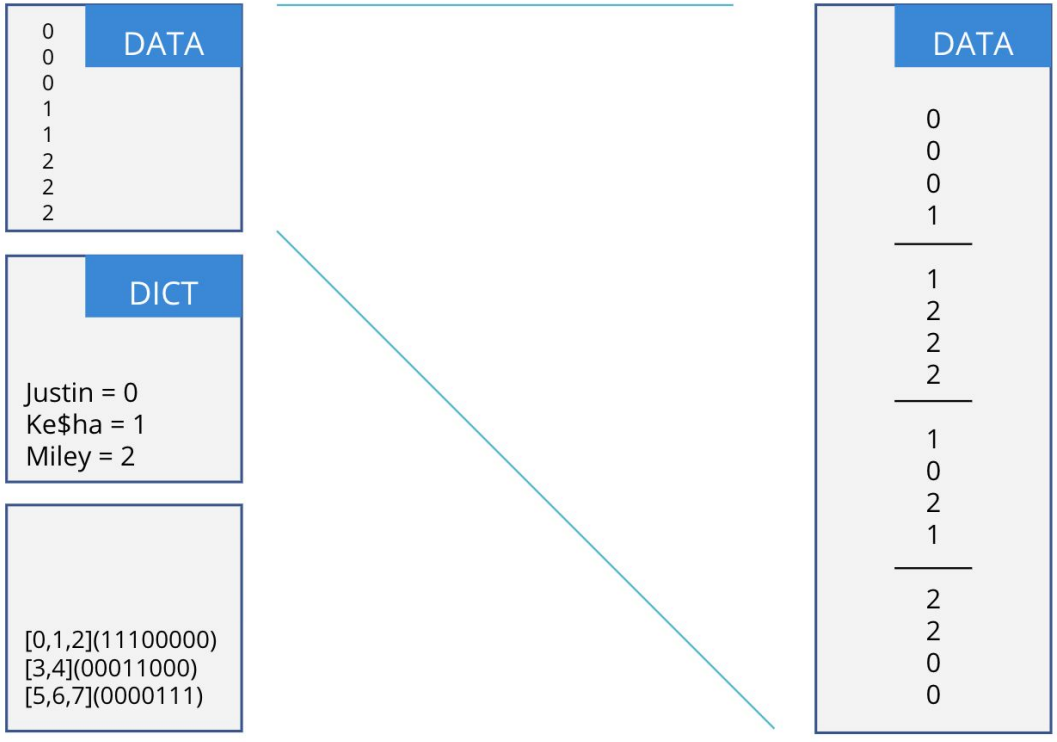


Mastering data layout

VIRTUAL
DRUID SUMMIT

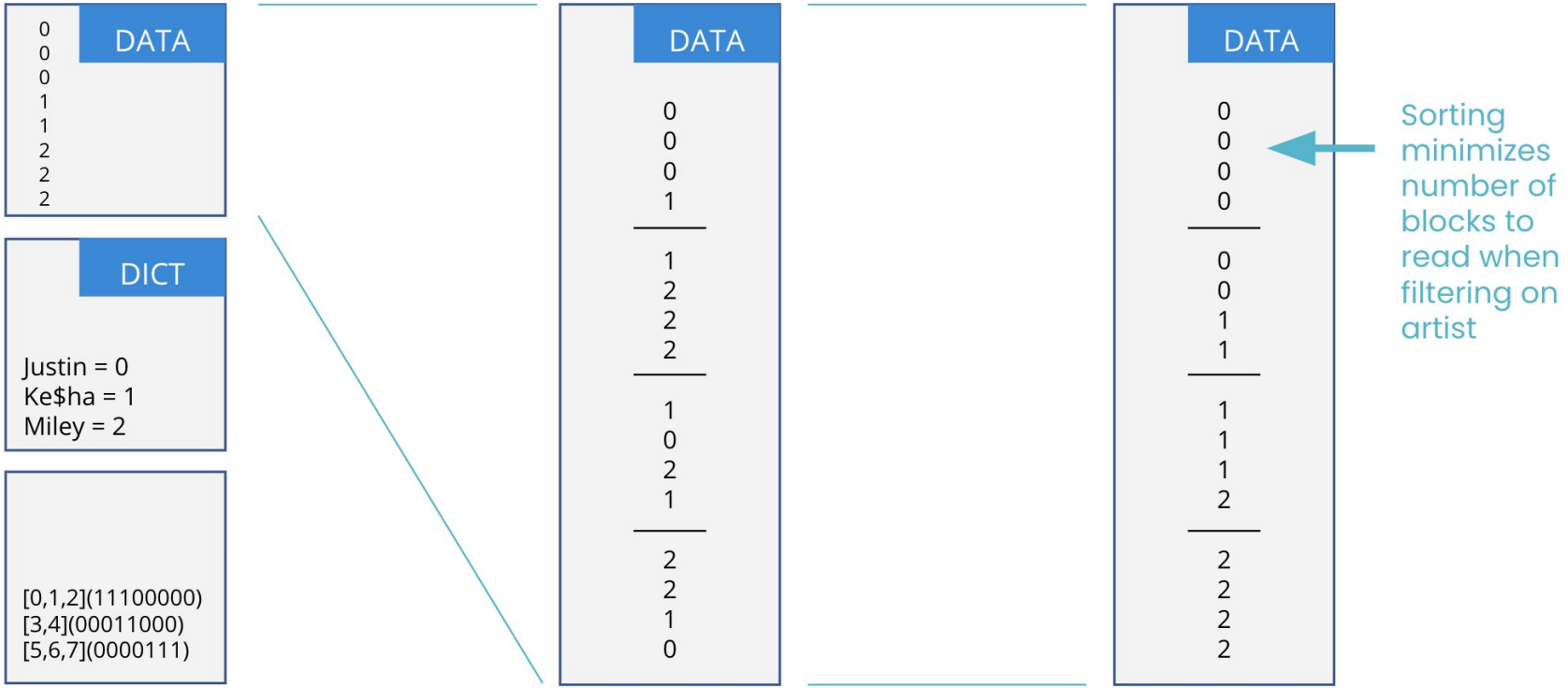
Mastering data layout

VIRTUAL
DRUID SUMMIT



Mastering data layout

VIRTUAL DRUID SUMMIT



Mastering data layout

VIRTUAL DRUID SUMMIT

```
“dimensionsSpec”: {
     “dimensions”: [
          “artist”,
          “city”
     ]
}
```

Rows are sorted first by time, then by dimensions in the order provided in the dimensionsSpec

Placing “artist” first puts it immediately after time in the sort order

Mastering data layout

VIRTUAL DRUID SUMMIT

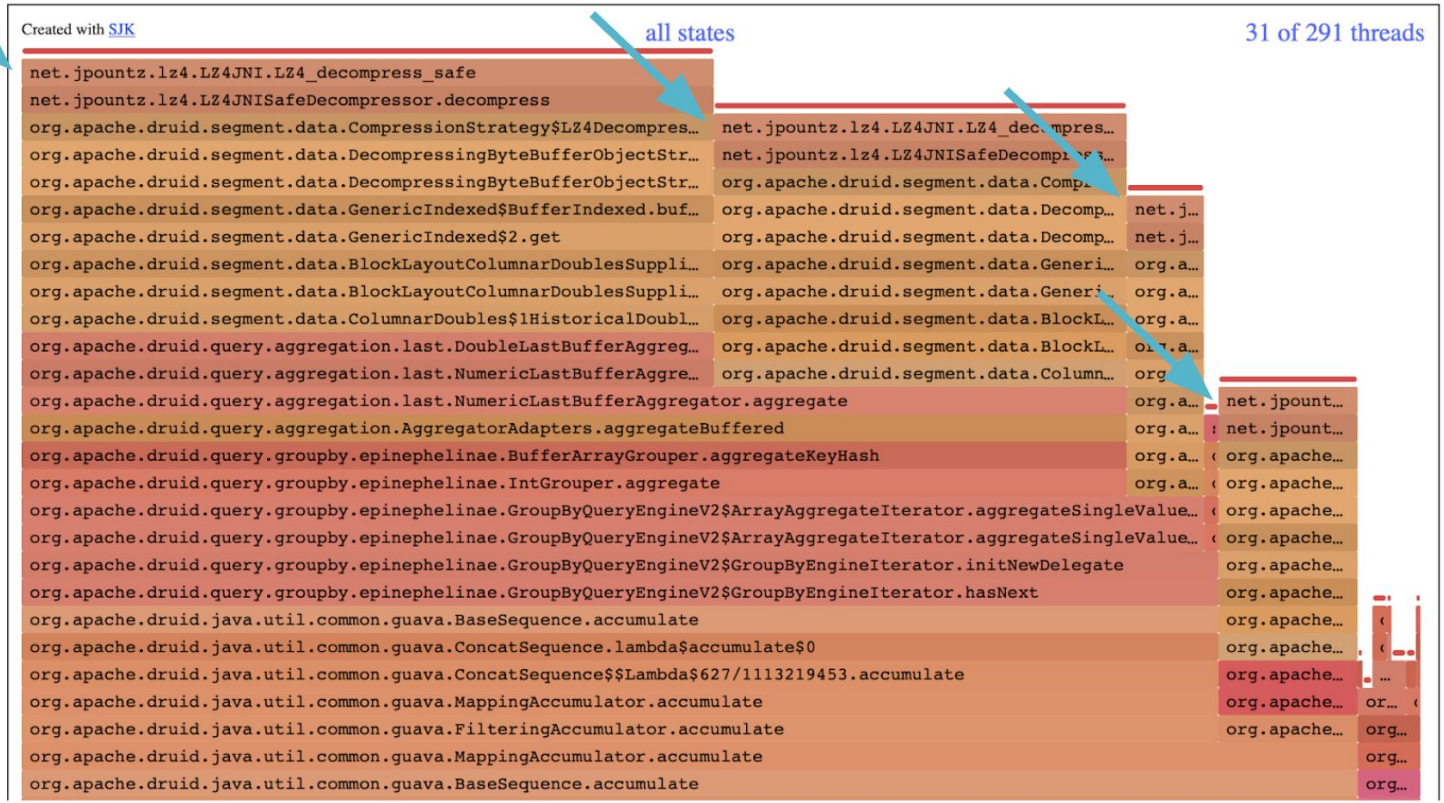
```
"dimensionsSpec": {
     "dimensions": [
          "artist",
          { "type" : "long", "name": "timestamp" },
          "city"
     ]
}

"timestampSpec": {
     "column": "timestamp",
     "format": "millis"
}

"granularitySpec": {
     "queryGranularity": "day",
     "segmentGranularity": "day"
}
```

Can put "artist" before time in the sort order using this *secondary timestamp* trick!

(Example assumes "timestamp" column in millis format.)

Mastering data layout

VIRTUAL DRUID SUMMIT



Mastering data layout

VIRTUAL
DRUID SUMMIT

Parachuting computations

Data size generally decreases in later stages of a group-by query

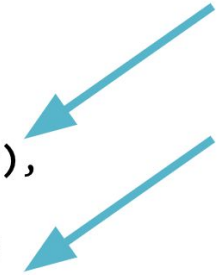
Move computations as late as possible

Parachuting computations

VIRTUAL DRUID SUMMIT

6 sec

```
SELECT
  LOWER(user),
  COUNT(*)
FROM traffic
GROUP BY 1
ORDER BY 2 DESC
```



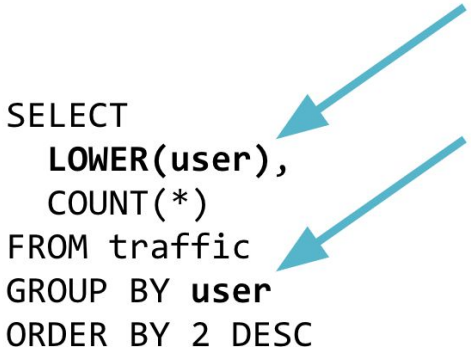
Grouping and selecting an expression of an underlying column

Parachuting computations

VIRTUAL DRUID SUMMIT

0.8 sec

(also 8x faster!)



```
SELECT
  LOWER(user),
  COUNT(*)
FROM traffic
GROUP BY user
ORDER BY 2 DESC
```

Group on column, select expression

Defers computation until after grouping

Not done automatically, because results will not be the same if the function maps two inputs to the same output

Stay in touch

VIRTUAL
DRUID SUMMIT

Follow the Druid project on Twitter!

@druidio

Join the community
(Mailing lists, Slack, meetups)
https://druid.apache.org/community/

Thank you!

Time for questions

@gianmerlino

VIRTUAL
DRUID SUMMIT