



PlazmaDB

ペタバイトオーダーのデータ分析
基盤を支える分散ストレージの
アーキテクチャとその運用

Who am I?

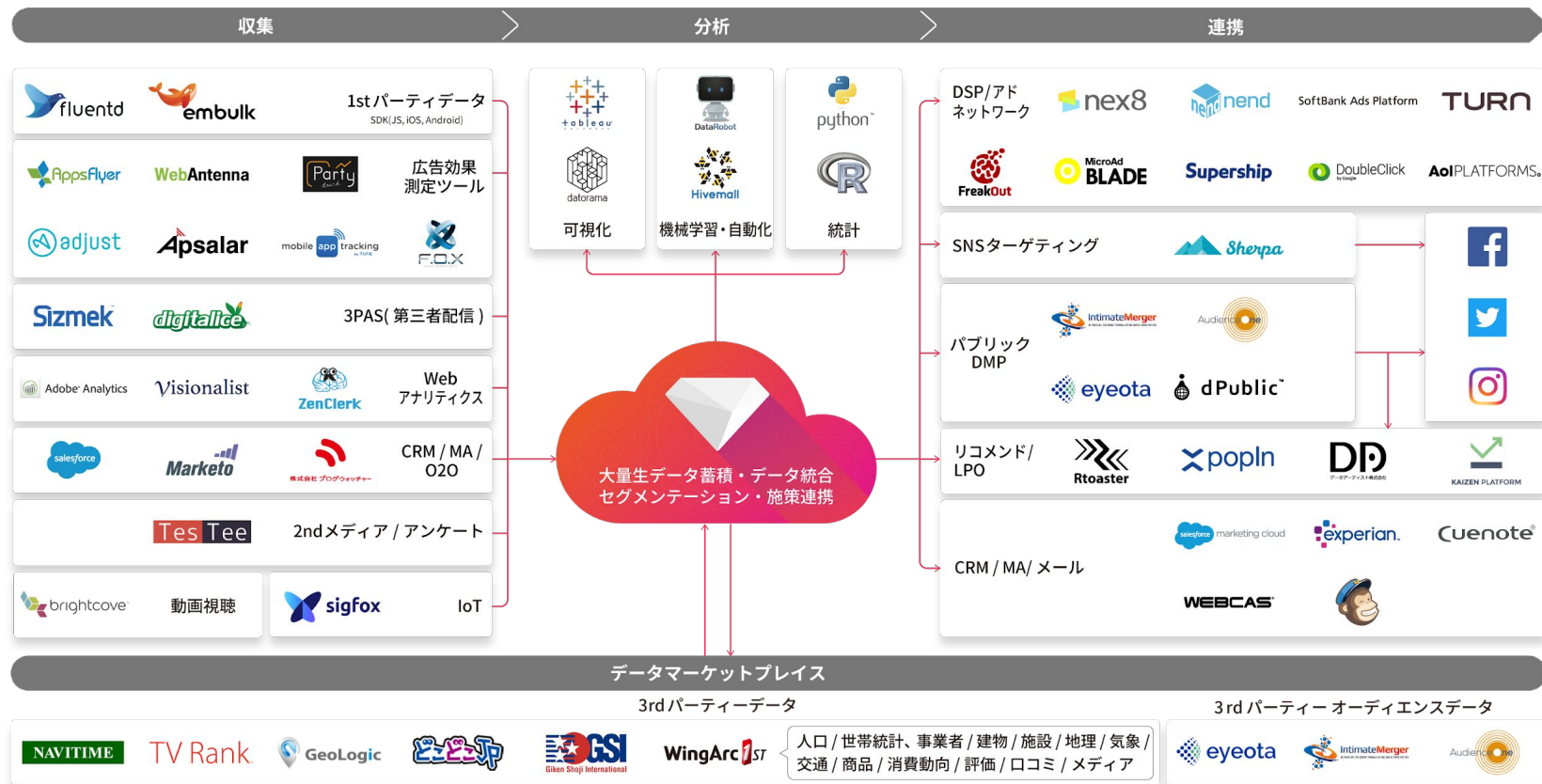
Keisuke Suzuki

- Backend Engineer @ Treasure Data KK
 - Ex. Fujitsu
- DB / Distributed system / Performance optimization
- Twitter: @yajilobee

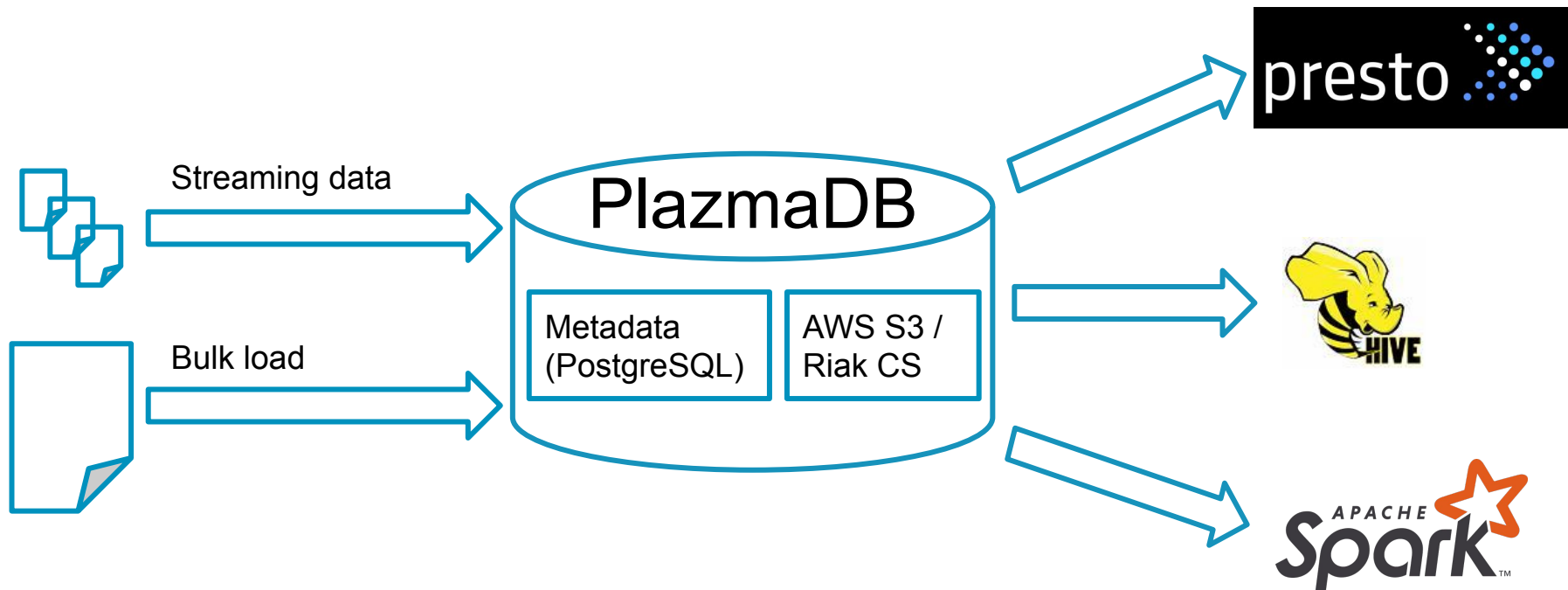


Treasure Data & PlazmaDB

Arm Treasure Data eCDP



PlazmaDB



Daily Workload & Storage Size

Import

500 Billion Records / day
~ 5.8 Million Records / sec

Query

600,000 Queries / day
15 Trillion Records / day

Storage size

5 PB (+5~10 TB / day)
55 Trillion Records

PlazmaDB Features

- Columnar format
 - Partitioned by time and optionally user defined column
- Schema less
- Partition index
- Partition optimization
 - Merge partitions
 - Realtime Storage & Archive Storage
- Transaction
 - Read committed isolation

PlazmaDB Features

- Columnar format
 - Partitioned by time and optionally user defined column
- Schema less
- Partition index
- Partition optimization
 - Merge partitions
 - Realtime Storage & Archive Storage
- Transaction
 - Read committed isolation

Log data

time	orderid	user	region	price	...
2018-01-01 10:00:00	1	1	'A'	10000	
2018-01-01 10:03:03	2	7	'C'	40000	
2018-01-01 10:23:03	3	6	'B'	3000	
2018-01-01 10:23:12	4	3	'A'	5500	
2018-01-01 11:04:44	5	1	'A'	20000	
2018-01-01 11:30:00	6	8	'C'	3000	
...					

Accumulate
over time

Many columns (attributes)

Analytical Query

time	orderid	user	region	price	...
2018-01-01 10:00:00	1	1	'A'	10000	
2018-01-01 10:03:03	2	7	'C'	40000	
2018-01-01 10:23:03	3	6	'B'	3000	
2018-01-01 10:23:12	4	3	'A'	5500	
2018-01-01 11:04:44	5	1	'A'	20000	
2018-01-01 11:30:00	6	8	'C'	3000	
...					

```
SELECT
```

```
  region,  
  SUM(price)
```

```
FROM
```

```
  orders
```

```
WHERE time >= '2018-01-01 10:00'
```

```
      AND time <= '2018-01-01 11:00'
```

```
GROUP BY
```

```
  region
```

Few part of columns

Filter by time window

Inefficiency of Row Based Format

time	orderid	user	region	price	...
2018-01-01 10:00:00	1	1	'A'	10000	
2018-01-01 10:03:03	2	7	'C'	40000	
2018-01-01 10:23:03	3	6	'B'	3000	
2018-01-01 10:23:12	4	3	'A'	5500	
2018-01-01 11:04:44	5	1	'A'	20000	
2018-01-01 11:30:00	6	8	'C'	3000	
...					

```
SELECT
  region,
  SUM(price)
FROM
  orders
WHERE time >= '2018-01-01 10:00'
      AND time <= '2018-01-01 11:00'
GROUP BY
  region
```

→ Scan direction



Scanned data

Columnar Format

time	orderid	user	region	price	...
2018-01-01 10:00:00	1	1	'A'	10000	
2018-01-01 10:03:03	2	7	'C'	40000	
2018-01-01 10:23:03	3	6	'B'	3000	
2018-01-01 10:23:12	4	3	'A'	5500	
2018-01-01 11:04:44	5	1	'A'	20000	
2018-01-01 11:30:00	6	8	'C'	3000	
...					



Scan direction



Scanned data

```
SELECT
```

```
  region,
```

```
  SUM(price)
```

```
FROM
```

```
  orders
```

```
WHERE time >= '2018-01-01 10:00'
```

```
      AND time <= '2018-01-01 11:00'
```

```
GROUP BY
```

```
  region
```

Few part of columns

Filter by time window

PlazmaDB Partitions

Table is collection of partitions

Application

{“time”: “2018-01-01 10:00:00”, “orderid”: 1, ...},
{“time”: “2018-01-01 10:03:03”, “orderid”: 2, ...}

{“time”: “2018-01-01 10:23:03”, “orderid”: 3, ...},
{“time”: “2018-01-01 10:23:12”, “orderid”: 4, ...}

{“time”: “2018-01-01 11:04:44”, “orderid”: 5, ...}

Send logs periodically

Worker

Convert to columnar
& Store a partition

PlazmaDB

2018-01-01 10:00:00	1	1	...
2018-01-01 10:03:03	2	7	...

2018-01-01 10:23:03	3	6	...
2018-01-01 10:23:12	4	3	...

2018-01-01 11:04:44	5	1	...
---------------------	---	---	-----

Partition file: A S3/RiakCS Object

PlasmaDB Metadata

PlasmaDB is Multi tenant

data_set_id: ID combination of User, Database, Table

Meta DB (PostgreSQL)

data_set_id	path	...
1		
1		
1		
2		

AWS S3 / RiakCS

Data set 1

2018-01-01 10:00:00	1	1	...
2018-01-01 10:03:03	2	7	...
2018-01-01 10:23:03	3	6	...
2018-01-01 10:23:12	4	3	...
2018-01-01 11:04:44	5	1	...

Data set 2

2018-01-01 10:00:00	1	1	...
2018-01-01 10:03:03	2	7	...

Partition Index

Meta DB (PostgreSQL)

data_set_id	time_range	path	...
1	[2018-01-01 10:00:00, 2018-01-01 10:03:03]		
1	[2018-01-01 10:23:03, 2018-01-01 10:23:12]		
1	[2018-01-01 11:04:44, 2018-01-01 10:04:44]		
2			

AWS S3 / RiakCS

Data set 1

2018-01-01 10:00:00	1	1	...
2018-01-01 10:03:03	2	7	...
2018-01-01 10:23:03	3	6	...
2018-01-01 10:23:12	4	3	...
2018-01-01 11:04:44	6	1	...

Data set 2

2018-01-01 10:00:00	1	1	...
2018-01-01 10:03:03	2	7	...

Partition Lookup

Meta DB (PostgreSQL)

data_set_id	time_range	path	...
1	[2018-01-01 10:00:00, 2018-01-01 10:03:03]		
1	[2018-01-01 10:23:03, 2018-01-01 10:23:12]		
1	[2018-01-01 11:04:44, 2018-01-01 10:04:44]		
2			

```
SELECT
  region,
  SUM(price)
FROM
```

```
  orders -- assume this is data set 1
WHERE time >= '2018-01-01 10:00'
      AND time <= '2018-01-01 11:00'
GROUP BY
  region
```


Skip Partition Scan

time	orderid	user	region	price	...
2018-01-01 10:00:00	1	1	'A'	10000	
2018-01-01 10:03:03	2	7	'C'	40000	
2018-01-01 10:23:03	3	6	'B'	3000	
2018-01-01 10:23:12	4	3	'A'	5500	
2018-01-01 11:04:44	5	1	'A'	20000	
2018-01-01 11:30:00	6	8	'C'	3000	
...					

→ Scan direction



Scanned data

```
SELECT
  region,
  SUM(price)
FROM
  orders
WHERE time >= '2018-01-01 10:00'
      AND time <= '2018-01-01 11:00'
GROUP BY
  region
```

How to find Partitions?

Meta DB (PostgreSQL)

data_set_id	time_range	path	...
1	[2018-01-01 10:00:00, 2018-01-01 10:03:03]		
1	[2018-01-01 10:23:03, 2018-01-01 10:23:12]		
1	[2018-01-01 11:04:44, 2018-01-01 11:04:44]		
2			

Number of partitions in a data set can be large (1M+) for large tables.

```
SELECT
  region,
  SUM(price)
FROM
```

```
  orders -- assume this is data set 1
WHERE time >= '2018-01-01 10:00'
      AND time <= '2018-01-01 11:00'
GROUP BY
  region
```

?

Range Type and GiST Index of PostgreSQL

Meta DB (PostgreSQL)

data_set_id	time_range	path	...
1	[2018-01-01 10:00:00, 2018-01-01 10:03:03]		
1	[2018-01-01 10:23:03, 2018-01-01 10:23:12]		
1	[2018-01-01 11:04:44, 2018-01-01 10:04:44]		
2			

GiST Index

Range type

- Overlap operator

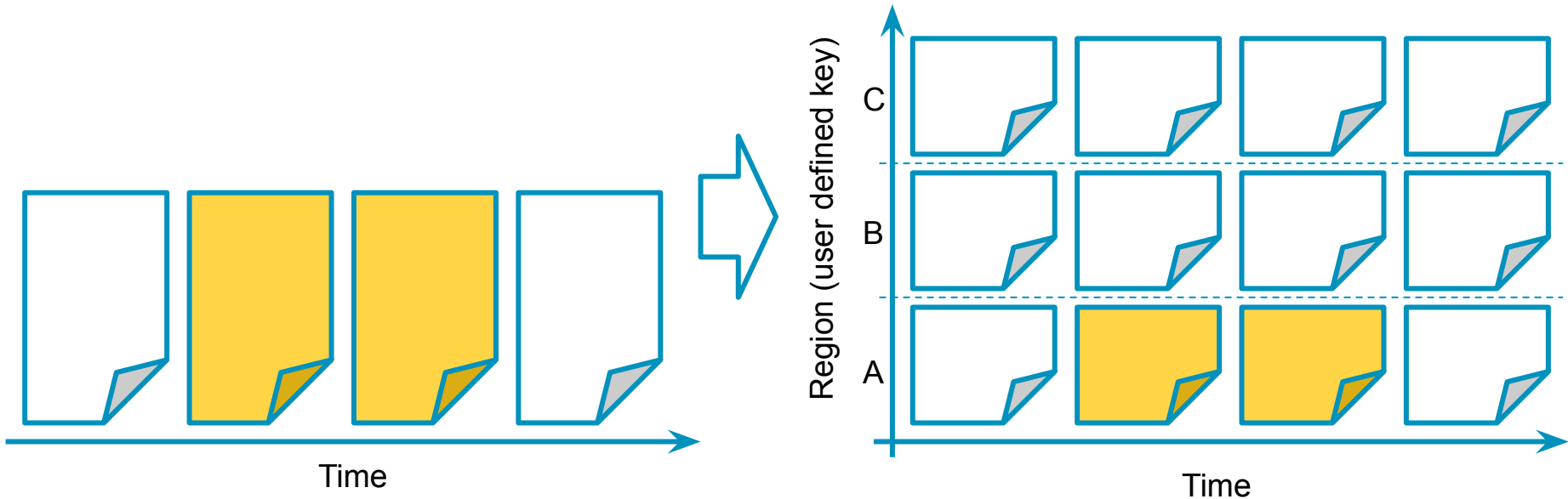
`time_range && [2018-01-01 10:00, 2018-01-01 11:00]`



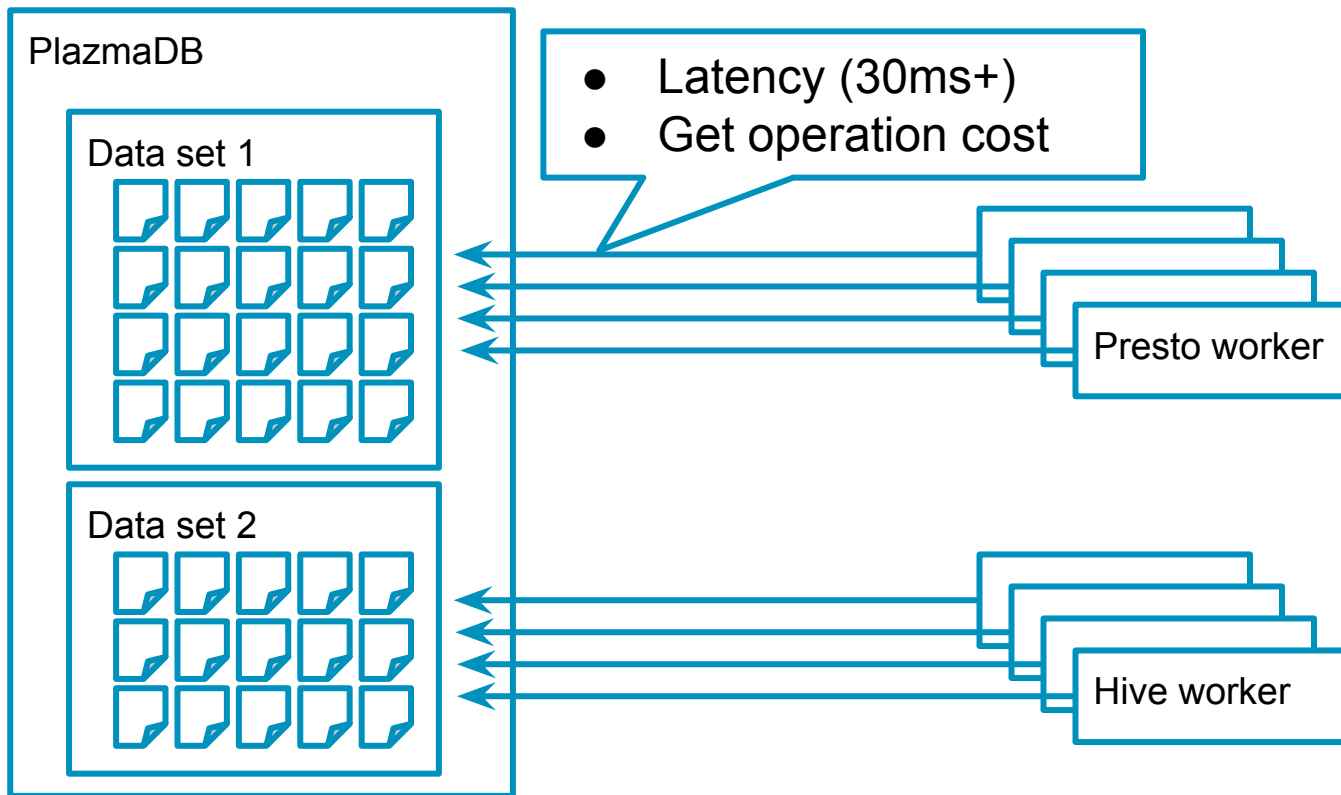
Overlap is checked by index scan

User Defined Partition (Beta)

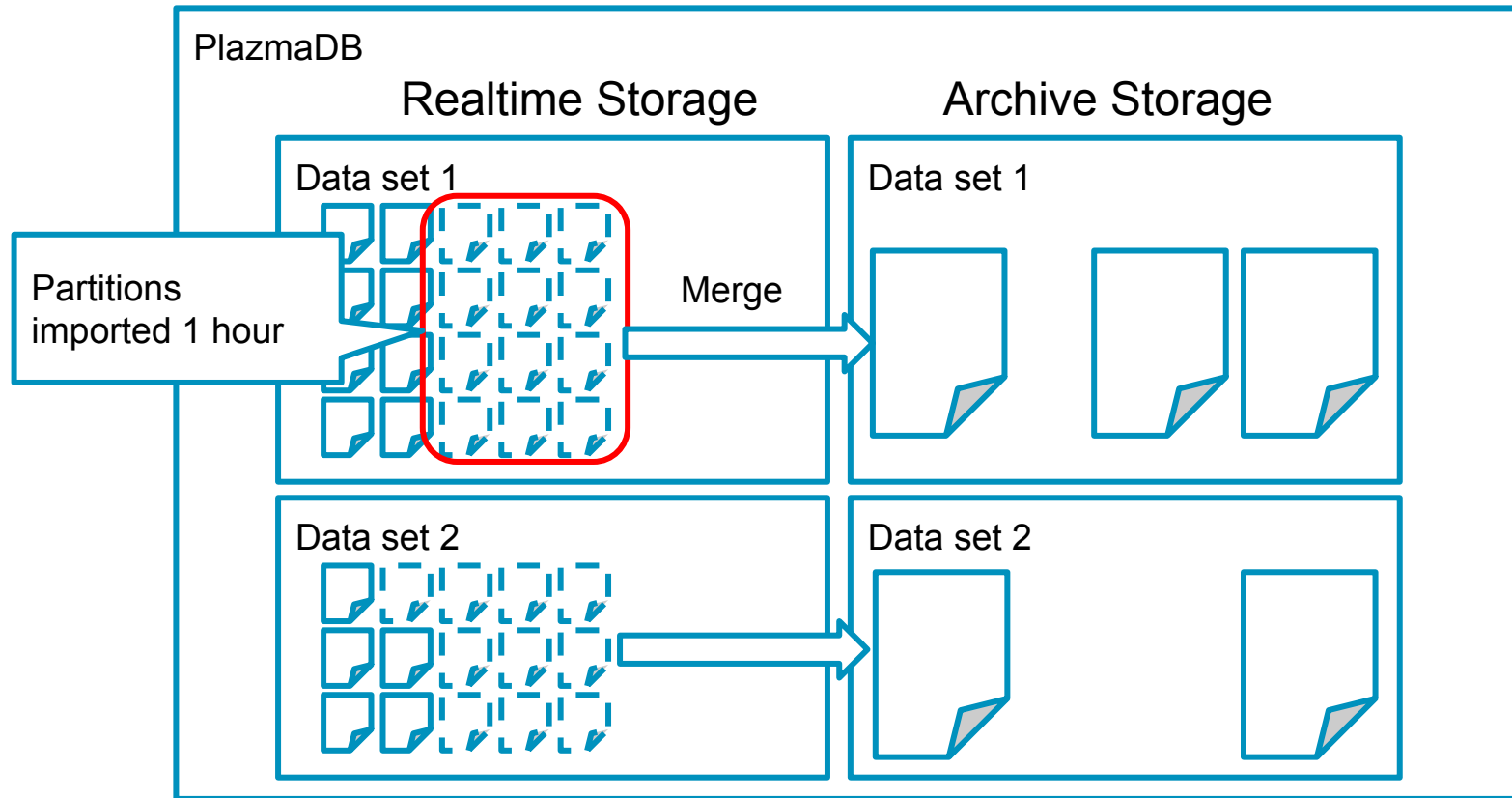
SELECT ... FROM ... WHERE time > ... AND time <... AND region = 'A'



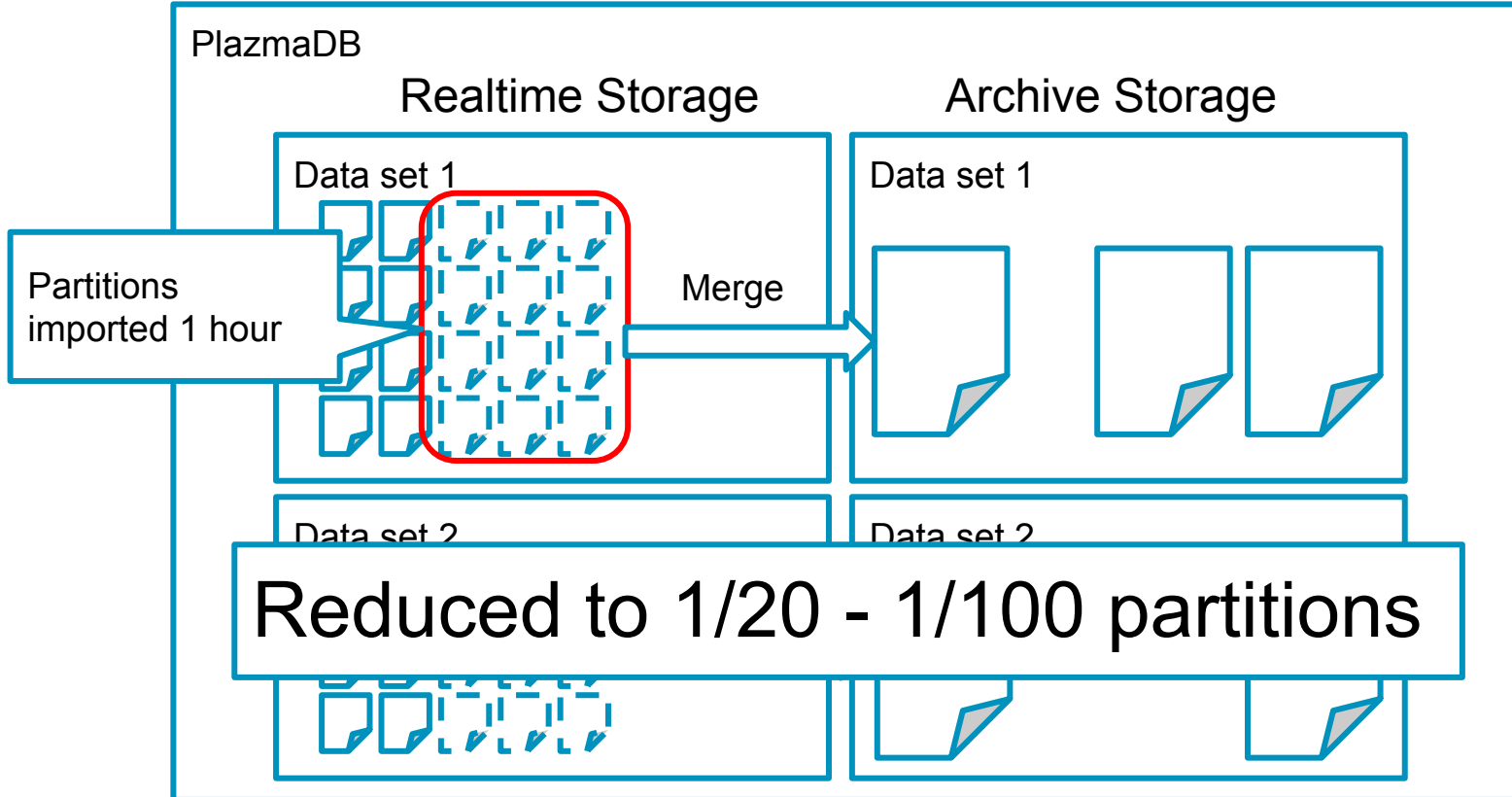
Partition Fragmentation



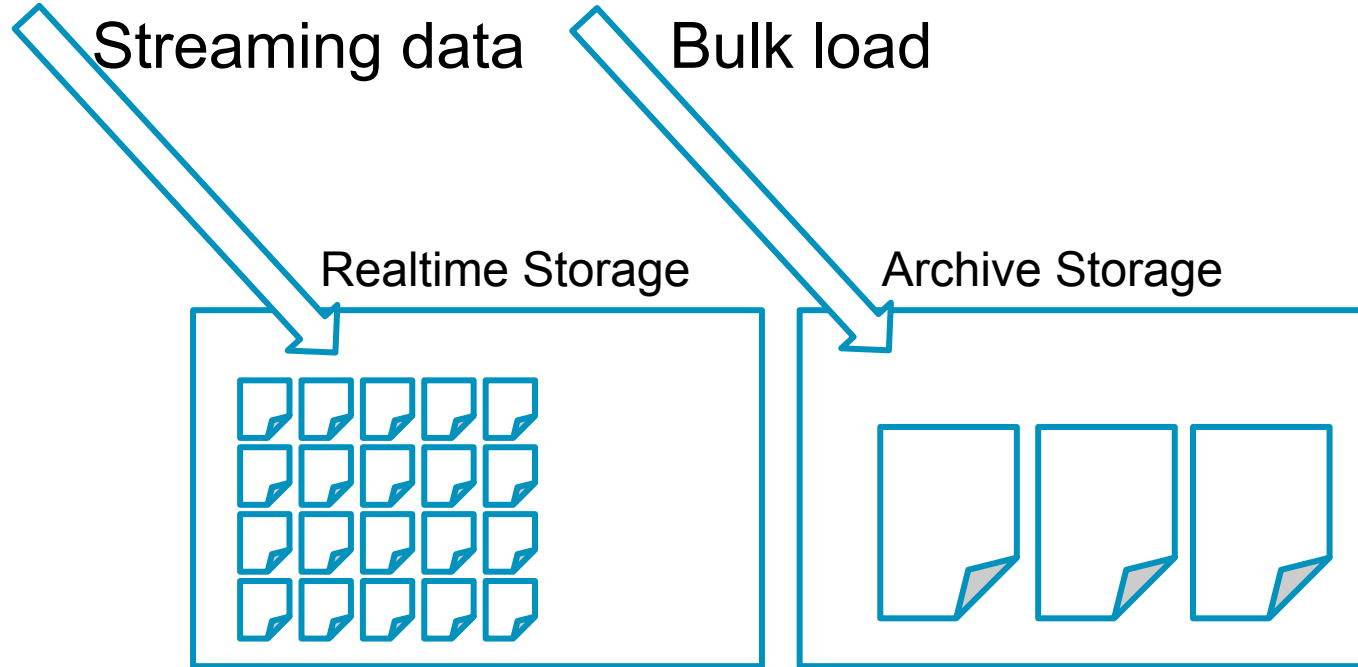
Realtime Storage & Archive Storage



Realtime Storage & Archive Storage



Streaming & Bulk data upload



Fragmentation of Archive Storage

Realtime Storage

data_set_id	time_range	import_time	...
1	[... 10:00:00, ... 10:03:03]	... 10:05:00	
1	[... 10:23:03, ... 10:23:12]	... 10:25:00	
...			
1	[... 20:30:00, ... 20:34:44]	... 20:35:00	
1	[... 10:00:44, ... 20:44:14]	... 20:45:00	
...			

Delayed data

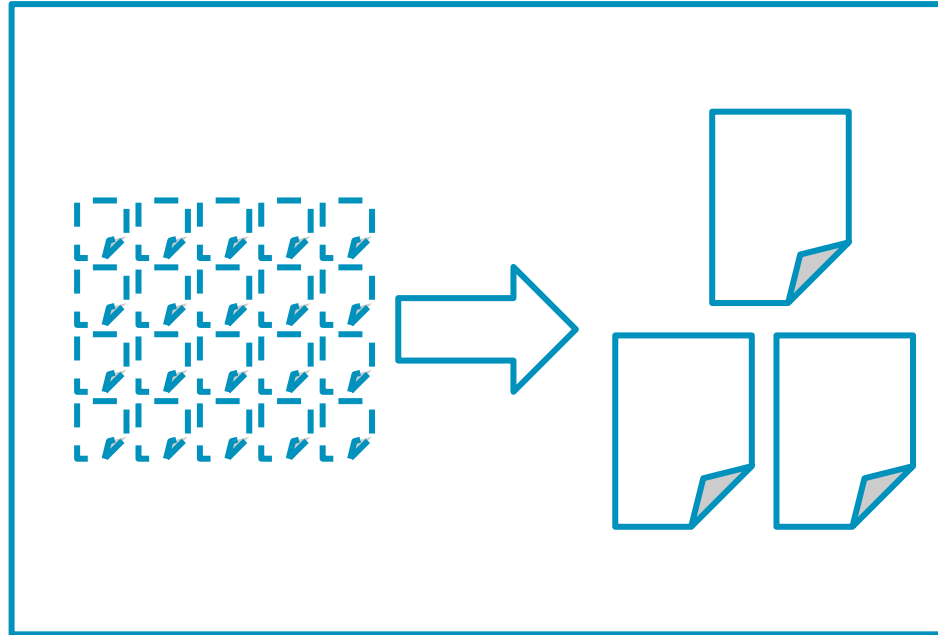
Archive Storage

data_set_id	time_range	...
1	[... 10:00:00, ... 11:00:00]	
...		
1	[... 10:00:44, ... 11:00:00]	
1	[... 15:04:38, ... 15:34:44]	
1	[... 20:00:00, ... 21:00:00]	
...		

Split by 1 hour window

Remerge Partitions

Archive Storage



Re: PlazmaDB features

- Columnar format
 - Partitioned by time and optionally user defined column
- Schema less
- Partition index
- Partition optimization
 - Merge partitions
 - Realtime Storage & Archive Storage
- Transaction
 - Read committed isolation

Current PlazmaDB & Future Challenges

Data Volume

PlazmaDB

Meta DB (PostgreSQL)

Realtime Storage

GiST

Partition
Metadata

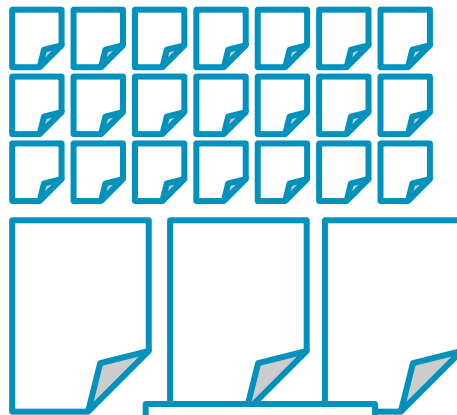
Archive Storage

GiST

Partition
Metadata

1 TB

AWS S3 / Riak CS



5 PB

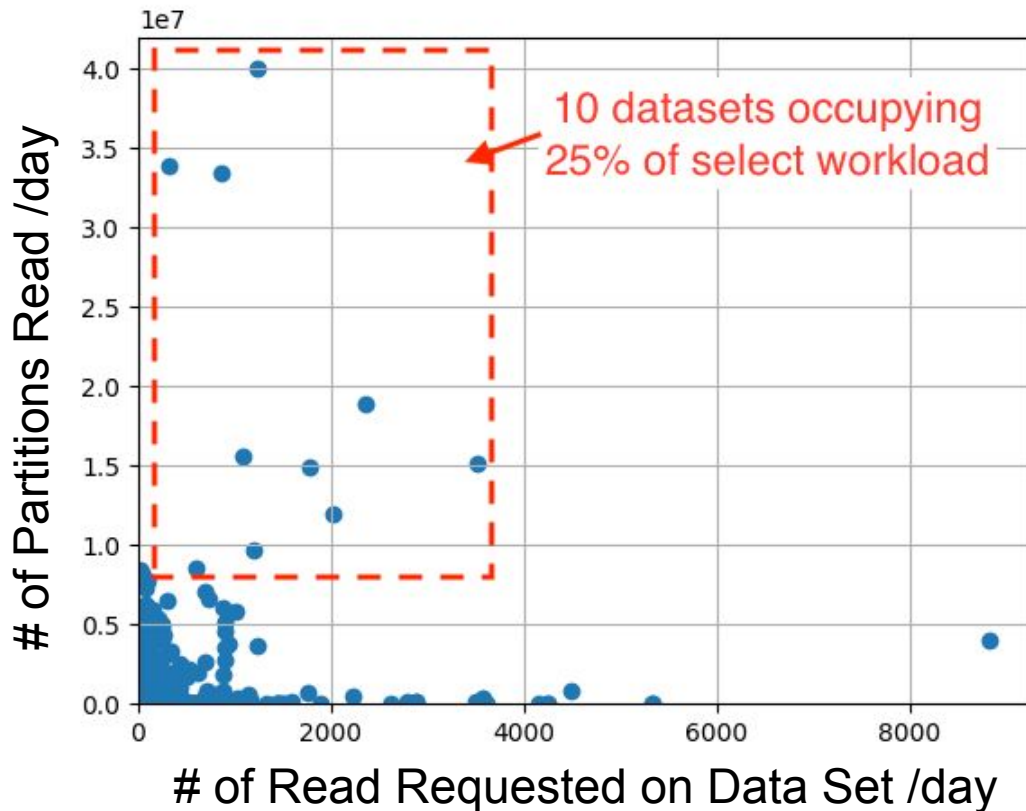
Monitoring

- Arm Treasure Data
 - Detailed log analyze
- DataDog
 - Metrics visualization



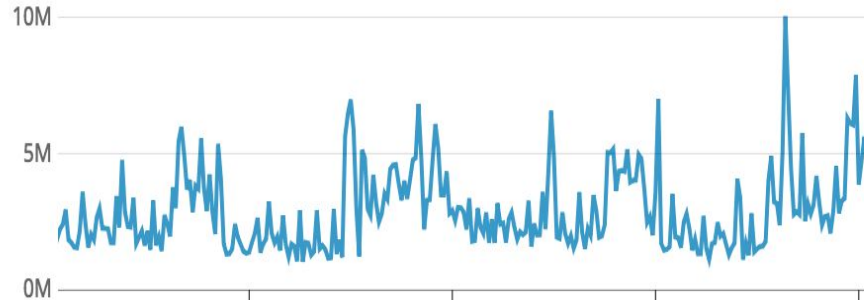
Read Workload on Meta DB

- Metadata size ~ 1TB
- Shared Buffer size ~ 150GB
- But, Hot Data size is much smaller than Shared Buffer



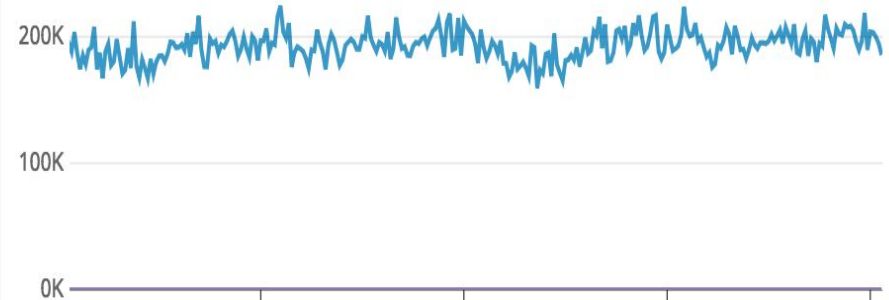
Write Workload on Meta DB

plazmadb WAL generation



3 MB / sec

DB: Transactions



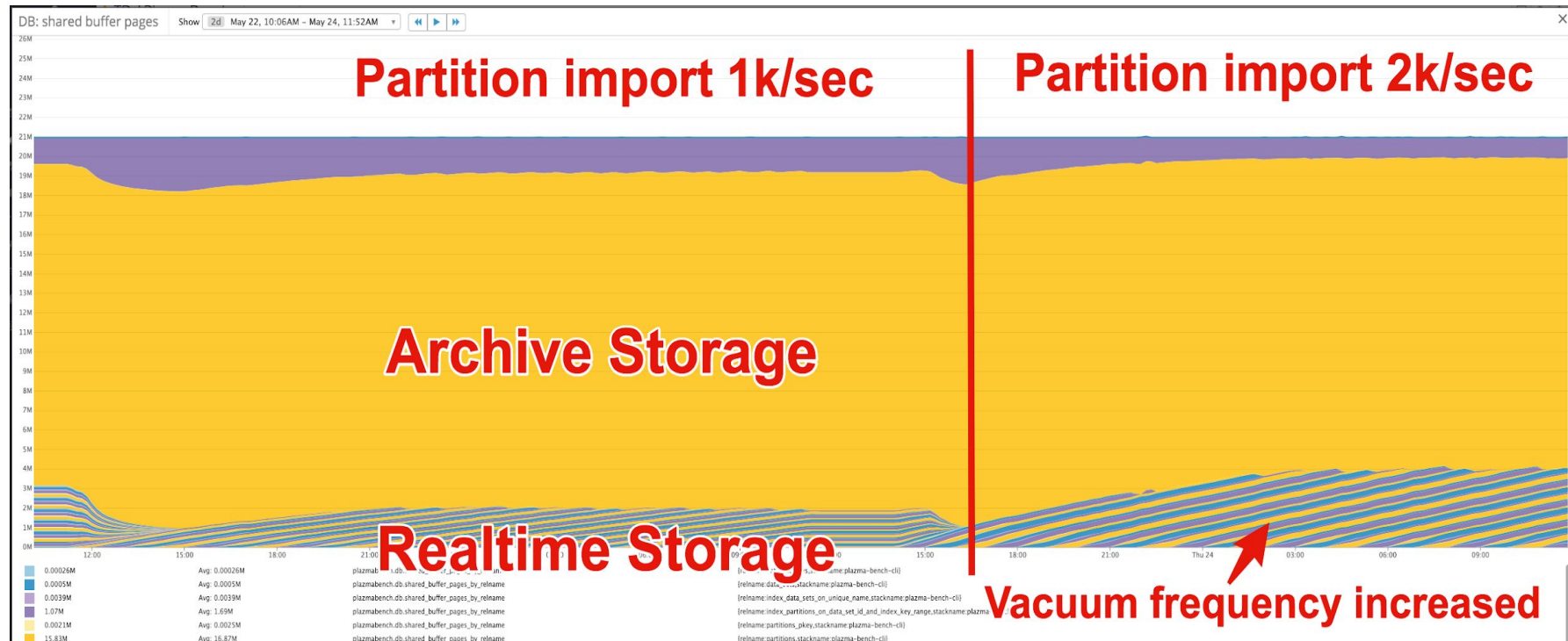
200k transaction / min
~ 3k transaction / sec

PostgreSQL Auto VACUUM (FREEZE)

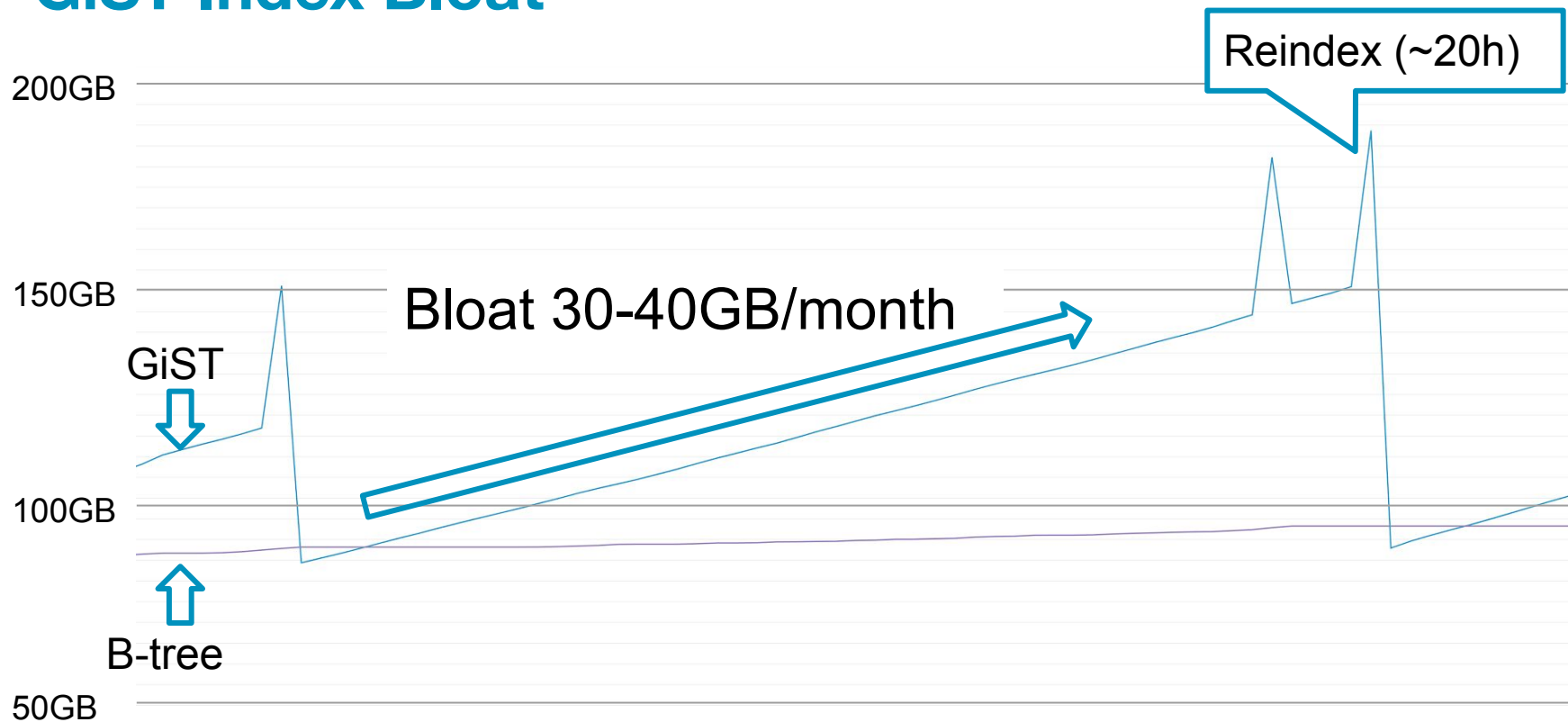
VACUUM FREEZE: Vacuum to prevent transaction ID wraparound failures

- Force full scan on relation (as of PostgreSQL 9.4)
 - Hot data may be evicted to scan relations for vacuum
=> **Read workload can be affected**
 - PostgreSQL 9.6 or later mitigate the problem
- The more transaction IDs are consumed, the more vacuum can be happened
 - In our case, it happens every 2-3 day

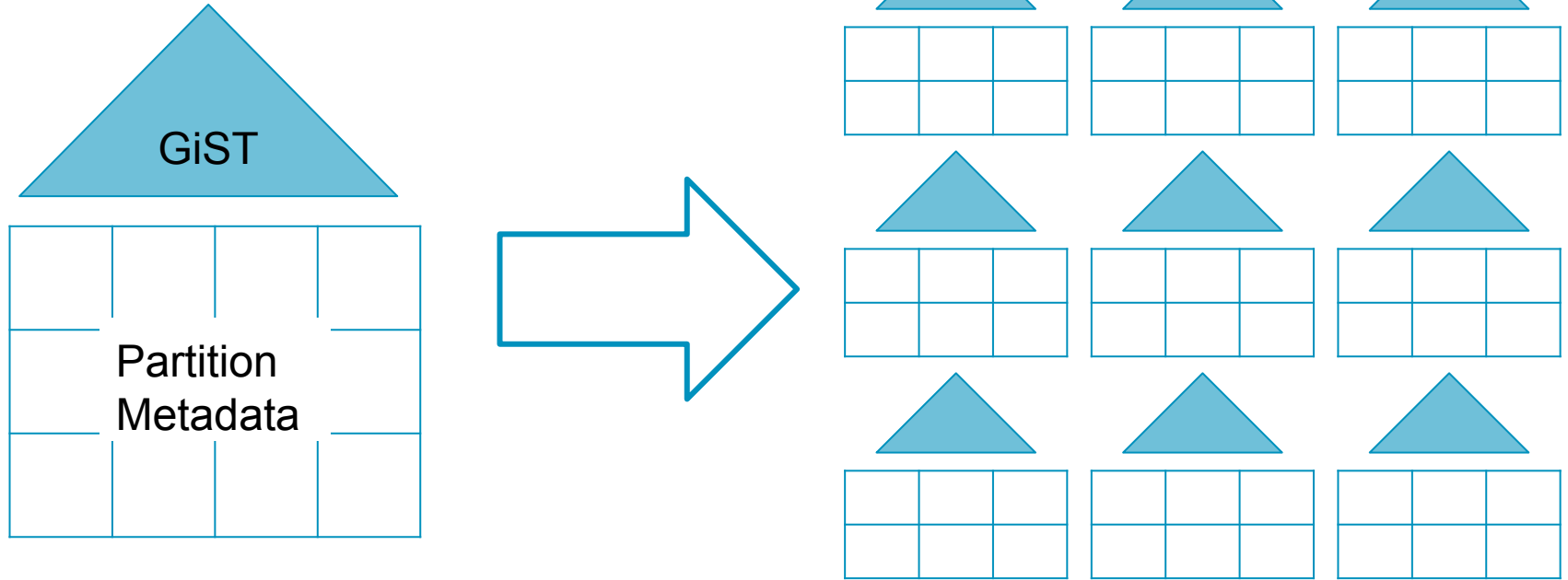
Impact of VACUUM FREEZE



GiST Index Bloat

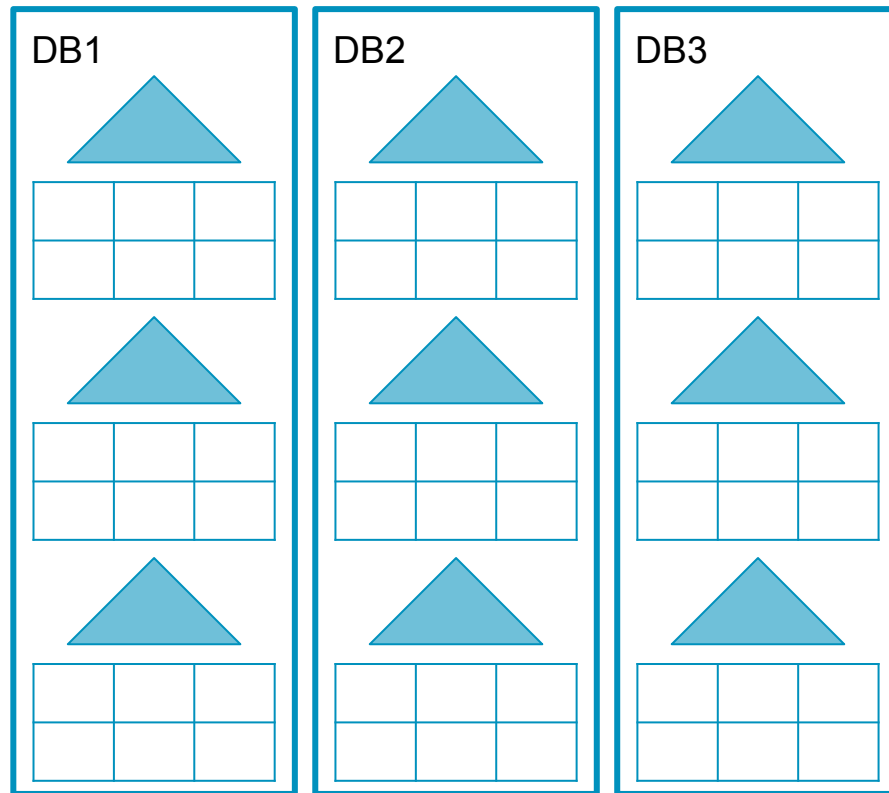


Metadata Table Partitioning



One relation's reindex becomes shorter and space saving

Meta DB Scale out



More Partition Skip

Meta DB (PostgreSQL)

data_set_id	time_range	path	...
1	[2018-01-01 10:00:00, 2018-01-01 10:03:03]		
1	[2018-01-01 10:23:03, 2018-01-01 10:23:12]		
1	[2018-01-01 11:04:44, 2018-01-01 10:04:44]		
2			

```
SELECT  
  region,  
  SUM(price)  
FROM
```

```
  orders  
WHERE time >= '2018-01-01 10:00'  
      AND time <= '2018-01-01 11:00'
```

```
      AND user_age >= 20  
      AND user_age <= 30
```

```
GROUP BY  
  region
```

Scan partition & Filter

More Partition Skip

Meta DB (PostgreSQL)

data_set_id	time_range	user_age_range
1	[2018-01-01 10:00:00, 2018-01-01 10:03:03]	[35, 40]
1	[2018-01-01 10:23:03, 2018-01-01 10:23:12]	[25, 30]
1	[2018-01-01 11:04:44, 2018-01-01 10:04:44]	
2		

```
SELECT  
  region,  
  SUM(price)  
FROM
```

```
  orders  
WHERE time >= '2018-01-01 10:00'  
      AND time <= '2018-01-01 11:00'  
      AND user_age >= 20  
      AND user_age <= 30
```

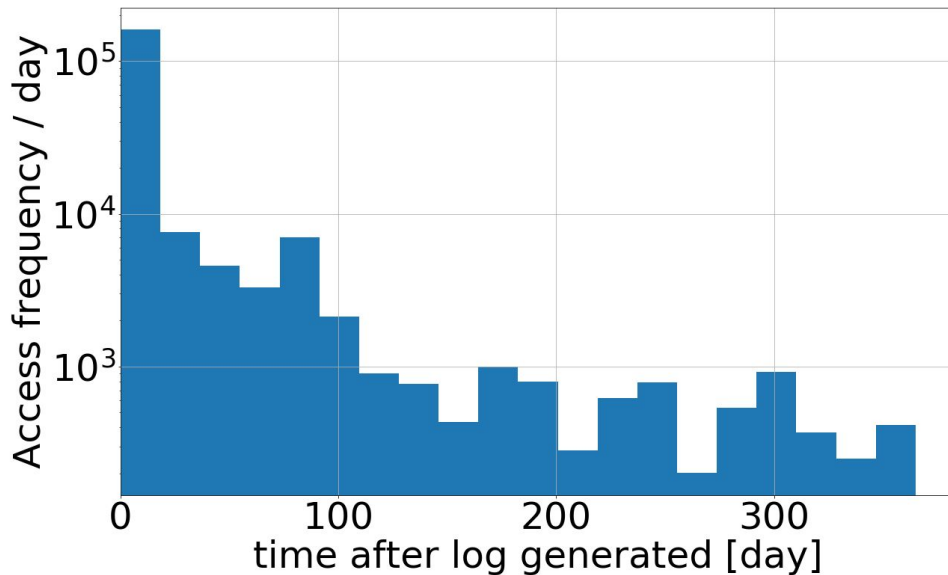
```
GROUP BY  
  region
```

Store metadata on frequently accessed columns

Smart Partition Selection for Remerge

Remerge is resource consuming

- Current
 - Data set size
 - # of Partitions
 - Idea
 - Access frequency
 - Data freshness
- Fresh data is likely to be hot



Summary

- PlazmaDB: Storage Layer of Arm Treasure Data Analytics Platform
 - Optimization for Analytical Queries
 - Columnar + Time Partitioning + Partition Index
 - Optimization for Streaming data
 - Realtime & Archive Storage + Merge Partition
- Challenges
 - Reduce impact of PostgreSQL VACUUM FREEZE
 - GiST index management
 - More Partition Optimization
 - Enrich Metadata
 - Smart Remerge

The background is a blue grid with small white dots. Overlaid on this are three solid-colored rectangles: an orange rectangle at the top center, a green rectangle on the right side, and a yellow rectangle at the bottom center.

We are Hiring!!

<https://www.treasuredata.com/company/careers/>

Thank You!

Danke!

Merci!

谢谢!

Gracias!

Kiitos!

