A journey toward the columnar data store

2025年5月14日

PGConf.dev 2025

About this Presentation

FUJITSU

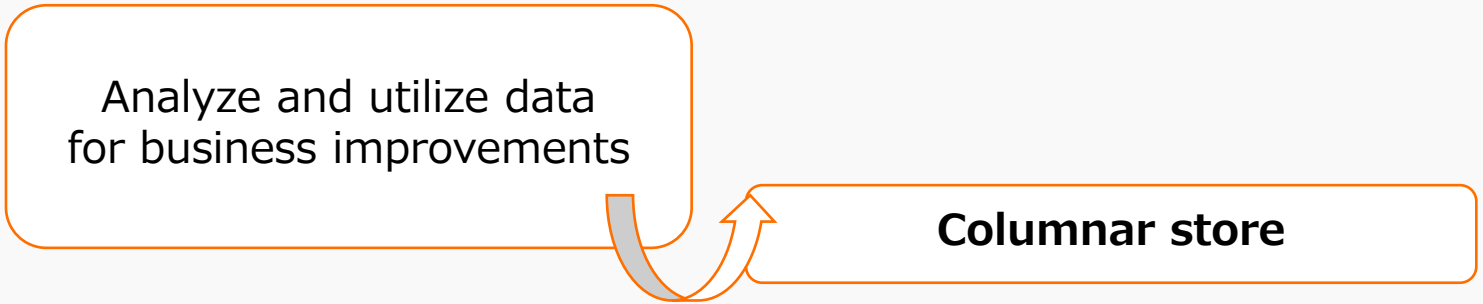
- Looking for people to work with us on this feature

Thread Title:
[WIP]Vertical Clustered Index (columnar store extension) - take2
-- We posted PoC patchs on this thread!

Business data utilization

FUJITSU

- EC: Optimize inventory based on sales data
- Financial Services: Transaction monitoring, fraud identification
- Manufacturing: Performance monitoring, anomaly detection, and maintenance timing prediction
- Transportation: Optimize delivery routes and alert you of risks



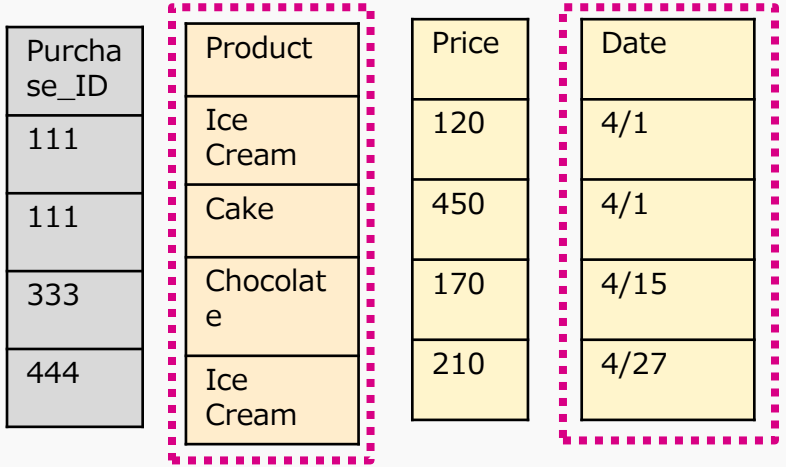
Columnar Store

- High-performance data retrieval
  - I/O can be reduced by scanning only columns needed for data analysis
  - All columns must be read in a row-oriented database

EX. SELECT Product, Date FROM purchase_history;

Purchase_ID	Product	Price	Date
111	Ice Cream	120	4/1
111	Cake	450	4/1
333	Chocolate	170	4/15
444	Ice Cream	210	4/27

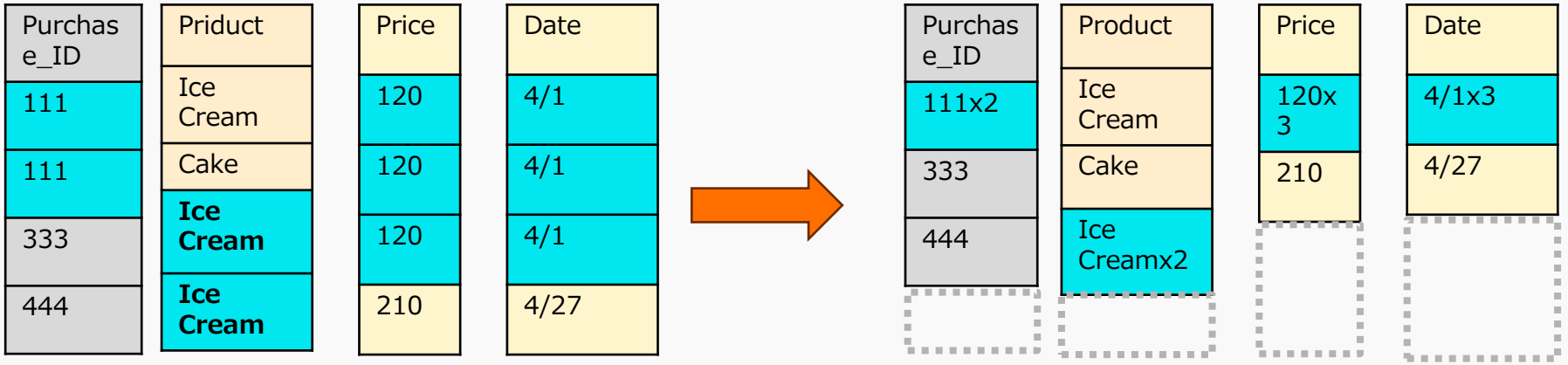
Need to read all columns



Scan only needed columns

Columnar Store

- Efficient compression
  - Long-term database can be effectively compressed.



Compression ratio is high because same patterns are repeated

PostgreSQL and columnar store

FUJITSU

Using PostgreSQL with data generated in an application ...

- PostgreSQL is a row-oriented database
- For high-performance data retrieval
  - Columnar Store
  - in-memory

- Pioneers of columnar store extensions
  - Citus (also available in Azure Cosmos DB for PostgreSQL)
  - Hydra
  - pg_mooncake

... but bad performance for **updates**

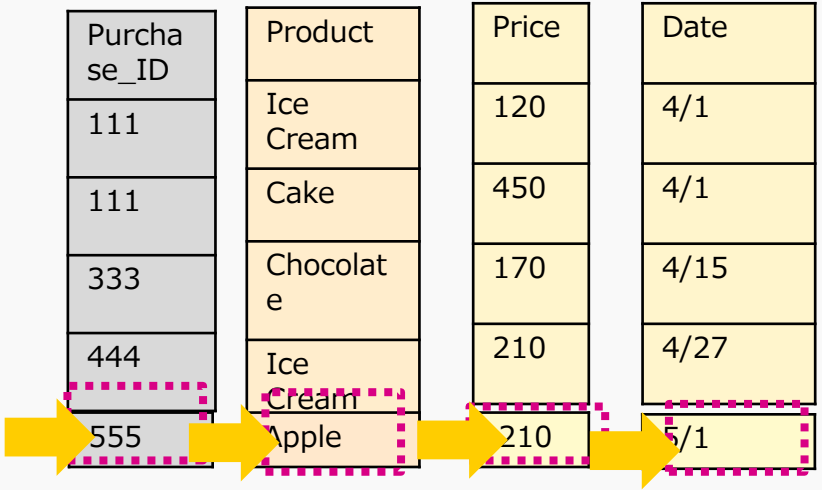
Columnar Store

- Not optimized for INSERT/UPDATE/DELETE
  - All columns must be read and updated

EX. INSERT INTO purchase_history VALUES (555, 'Apple', 210, ‘5/1');

Purchase_ID	Product	Price	Date
111	Ice Cream	120	4/1
111	Cake	450	4/1
333	Chocolate	170	4/15
444	Ice Cream	210	4/27
555	Apple	210	5/1

Done at once



Needs I/O per attributes

Real-time utilization of business data

FUJITSU

Data analysis you want to perform:

- EC: Optimize inventory based on sales data
- Financial Services: Transaction monitoring, fraud identification
- Manufacturing: Performance monitoring, anomaly detection, and maintenance timing prediction
- Transportation: Optimize delivery routes and alert you of risks

Data generated:

- EC: Insert purchase history and update inventory data
- Financial Services: Insert Transaction History and Balance Update
- Manufacturing: Insert sensor information by hour
- transportation: updating location and dispatch information

→ Update performance must not be slow

→ Need to be able to analyze with updated data

in information analysis
With data generated in the business
real-time utilization

Not slow down the updating

Analysis with the latest data

retrieval performance of Columna Store

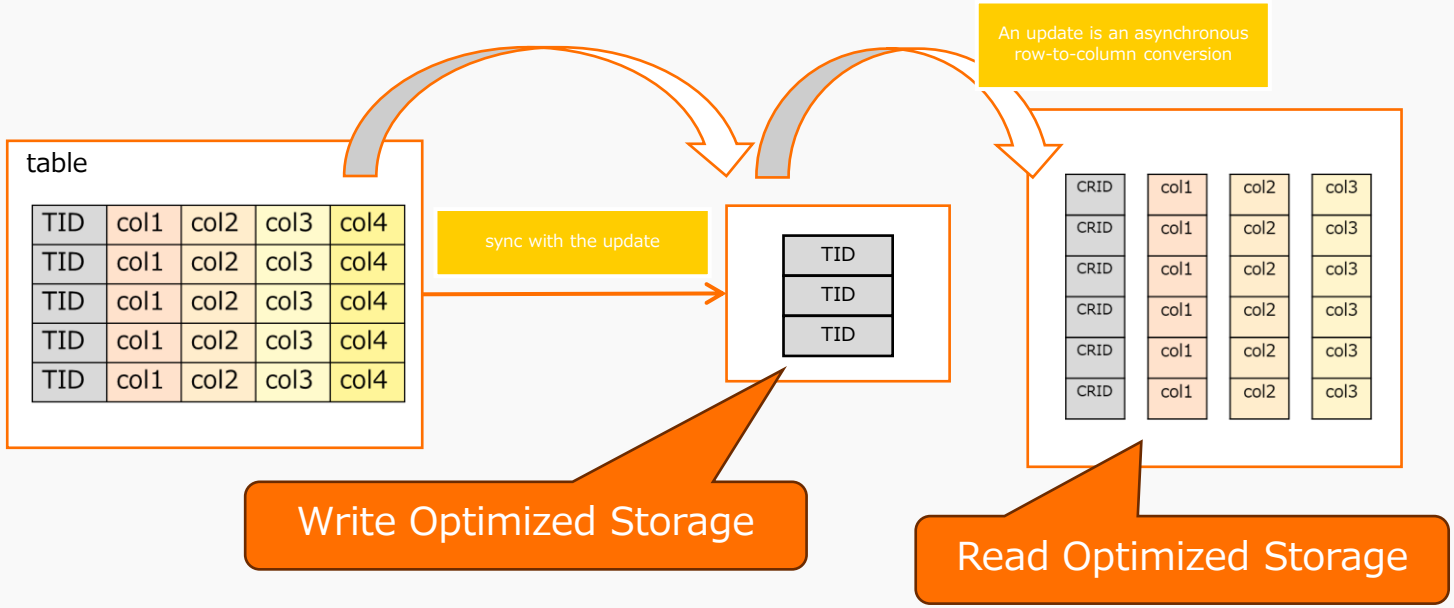
VCI

Vertical Clustered Index (VCI) Proposal

FUJITSU

- Columnar store function using column type index
- I am currently developing a prototype.
- VCI allows you to:
  - **Ability to aggregate data generated by OLTP in real time**
  - Faster data retrieval
- Current functional range of VCI
  - Columnar Store
    A) Conversion to column is asynchronous to avoid degradation of update performance
    B) Refer to all data including update data using column format data and update difference information
  - data compression
  - parallel scan
  - In-memory

How VCI Works



1. INDEX Creation

FUJITSU

- CREATE INDEX creates **Read Optimized Storage** for the specified row

Purchase History Table

Purchase_ID	Product	Price	Date
111	Ice Cream	120	4/1
111	Cake	450	4/1
333	Chocolate	170	4/15
444	Ice Cream	210	4/27

CREATE INDEX vcindex …
USING vci (Price, Date)…;

Price	Date
120	4/1
450	4/1
170	4/15
210	4/27

ROS

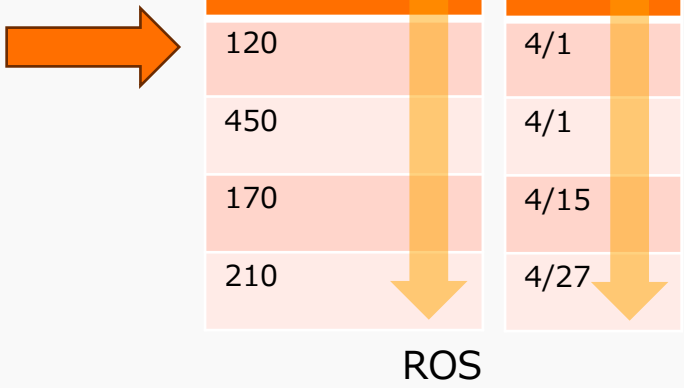
2. Searching Tables

FUJITSU

- Find ROS data when executing SELECT query
- the ability to quickly search only product lines

Ex. tally up sales of goods

```
SELECT SUM(price) FROM purchase_history
  WHERE date > '2025-03-31';
```



3. Updating Tables

- When INSERT occurs, store TID in **Write Optimized Storage** and ROS do not update
- Only inserts into small data to WOS, so update performance impact is low

TID
1
2
3
4

Purchase_ID	Product	Price	Date
111	Ice Cream	120	4/1
111	Cake	450	4/1
333	Chocolate	170	4/15
444	Ice Cream	210	4/27

INSERT;

555	Apple	198	5/1

TID
5

WOS

Price	Date
120	4/1
450	4/1
170	4/15
210	4/27

ROS

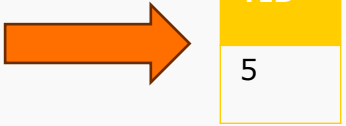
4. SELECT after INSERT

- Temporarily convert WOS data to **Local ROS**
- Search for Local ROS and ROS

```
SELECT SUM(price) FROM purchase_history
  WHERE date > '2025-03-31';
```

TID
1
2
3
4
5

Purchase_ID	Product	Price	Date
111	Ice Cream	120	4/1
111	Cake	450	4/1
333	Chocolate	170	4/15
444	Ice Cream	210	4/27
555	Apple	198	5/1



TID
5

WOS

Conversion

Price	Date
198	5/1

Local ROS

Price	Date
120	4/1
450	4/1
170	4/15
210	4/27

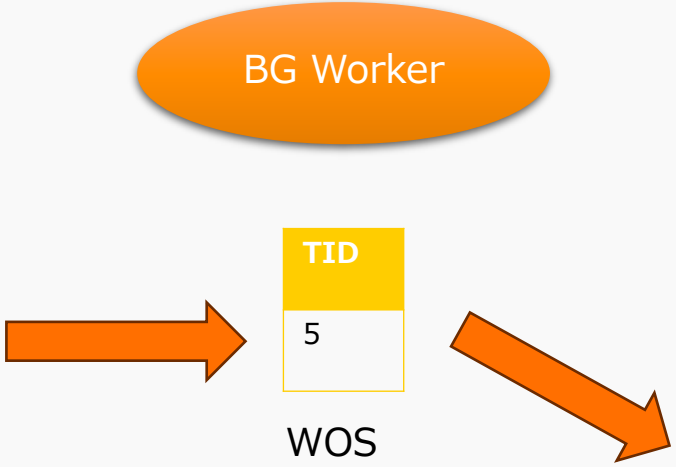
ROS

5. ROS Update (Asynchronous)

FUJITSU

BG Worker propagates changes to ROS after a while

ID	Merchandise	Price	Date of purchase
111	Ice Cream	120	4/1
111	Cake	450	4/1
333	Chocolate	170	4/15
444	Ice Cream	210	4/27
555	Apple	198	5/1



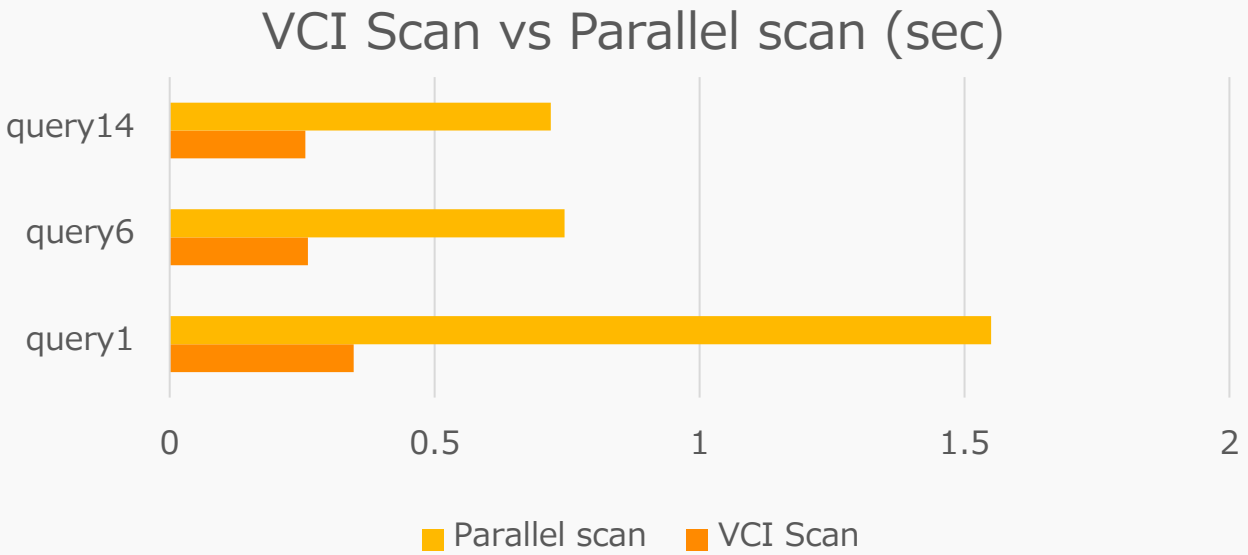
Price	Date
120	4/1
450	4/1
170	4/15
210	4/27
198	**5/1**

ROS

performance measurement

FUJITSU

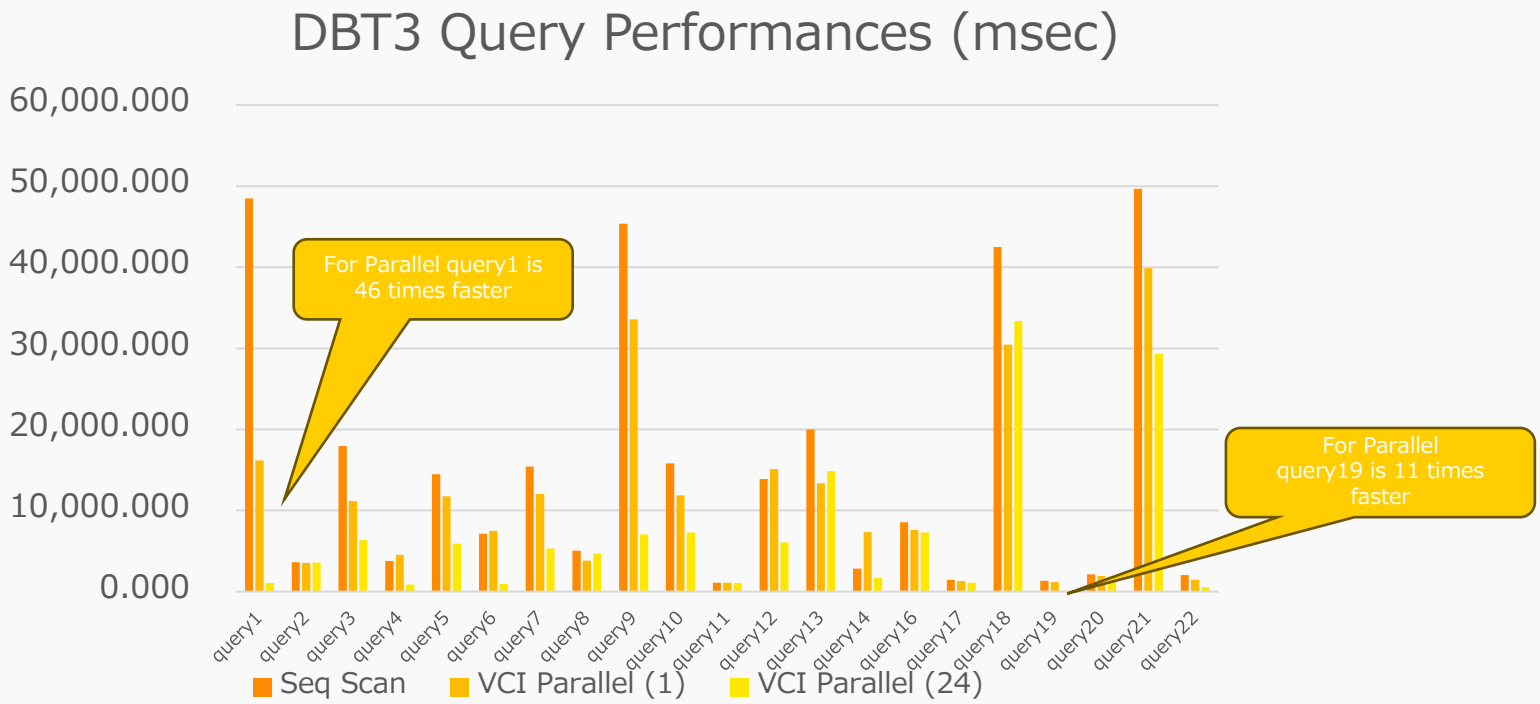
- TPC-H scans 4.4 times faster
  - NOTICE: It does not a measurement using posted patches



performance measurement

FUJITSU

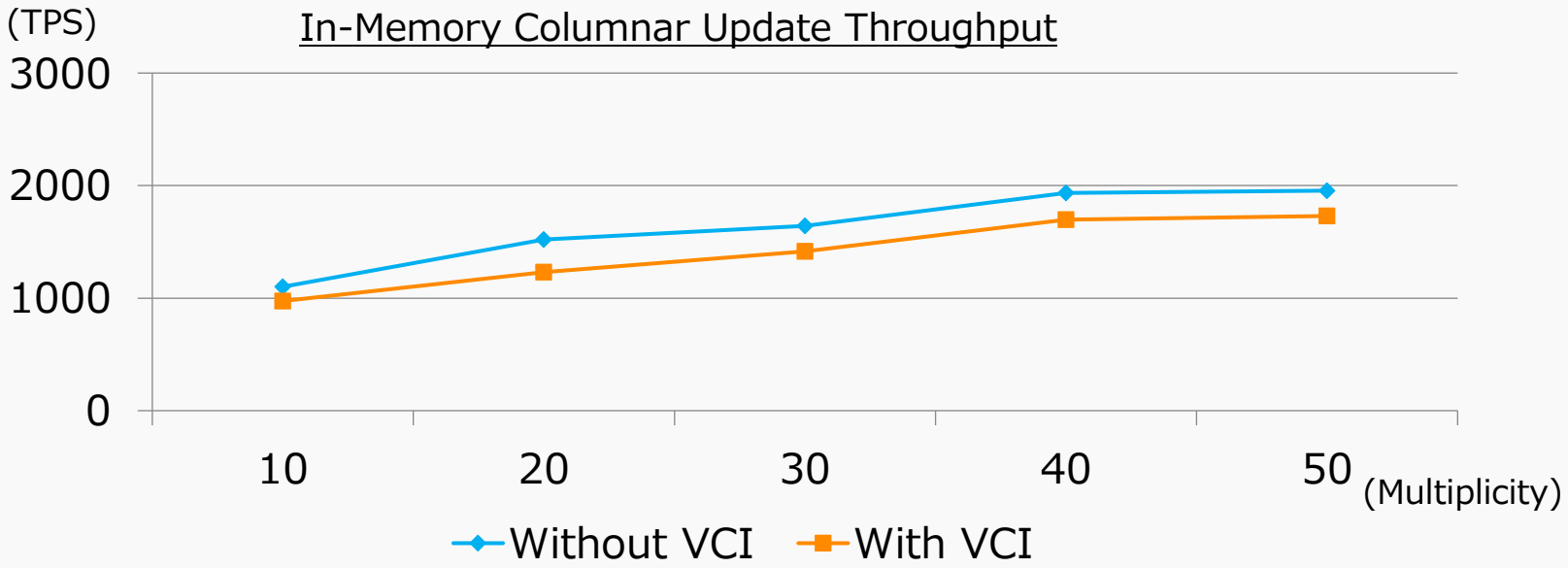
- DBT3 measurement (v 9.5)



performance measurement

FUJITSU

- The performance of update processing is almost the same with or without VCI



Results from our company measurement environment

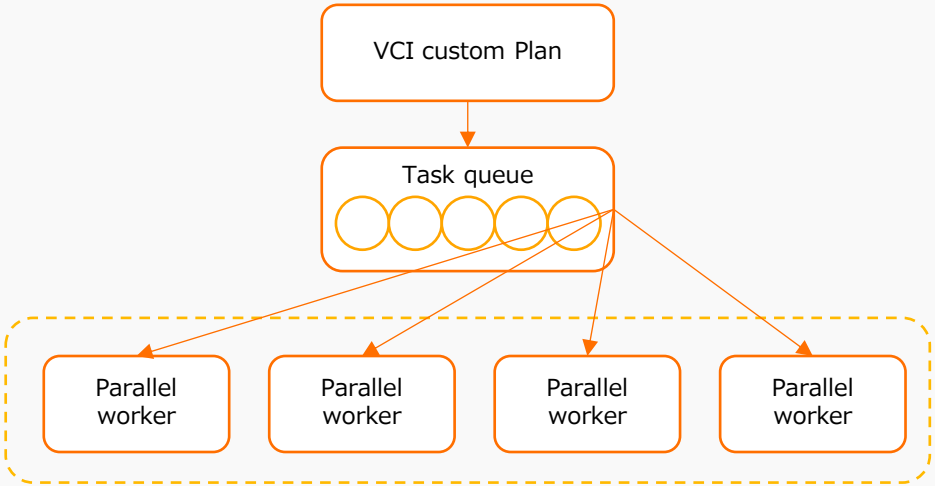
Current VCI Implementation

FUJITSU

- Using PostgreSQL mechanisms
  - Index Access Method
    - VCI uses PostgreSQL custom indexes
  - Executor hook
    - Use executor hook to replace with VCI plan
  - Custom Scan
    - Create custom plans for four types of VCI

VCI Implementation

- proprietary implementation
  - Custom implementation of parallel scanning with VCI
    - This requires changes to PostgreSQL's standard implementation
  - Implementing a new hook
    - Implement hooks for Index and Relation operations



Functions that are not implemented or lacking

FUJITSU

Missing items

- Limited datatypes are supported
  - Text search types and JSON types are not supported
- Needs time for setup
  - Time consuming to define indexes on existing data
- pg_upgrade support
  - Get VCI definitions
  - Drop extensions once
  - Install VCI and create indexes again after the upgrade

Responding to Hook

- The following hooks are additionally implemented
- DELETE execution hook (amdelete)
  - Remove columns from columns whenever VCI indexed tables are updated or deleted, not when VACUUM is triggered
- Change when aminsert is called
  - Change to call if columns not specified in CREATE INDEX column field are UPDATE

These undeveloped features
We are looking for members to develop together!

Thank you

FUJITSU