Let’s scale-out PostgreSQL using Citus

Noriyoshi Shinoda

November 22, 2018

Speaker

Noriyoshi Shinoda

✓Affiliated company

- Hewlett-Packard Enterprise Japan

✓Current work

- System design, tuning, consulting on PostgreSQL, Oracle Database, Microsoft SQL Server, Vertica, Sybase ASE, etc. related to RDBMS
- Oracle ACE
- Written 15 books related to Oracle Database
- Investigation and verification on open source products

✓URL

- Published documents
  http://slideshare.net/noriyoshishinoda/
- Oracle ACE
  https://apex.oracle.com/pls/apex/f?p=19297:4:::NO:4:P4_ID:2780

Agenda

✓What's Citus?

✓Let's try!

✓Architecture

✓Restriction

✓Behavior when trouble occurs

This slides is based on Citus Community Edition 8.0-8

What’s Citus?

What’s Citus?

What’s Citus?

✓ Achieves scale-out environment for PostgreSQL

- Parallel query and partitioning feature across multiple nodes

✓Implemented as PostgreSQL Extension

✓Developed by Citusdata

- Community Edition is open source
- To use features such as online rebalancing, Enterprise Edition is required

✓It does not include the following features

- Automatic failover
- Automatic data rebalance
- Operational features such as backup

What's Citus?

Instance configuration

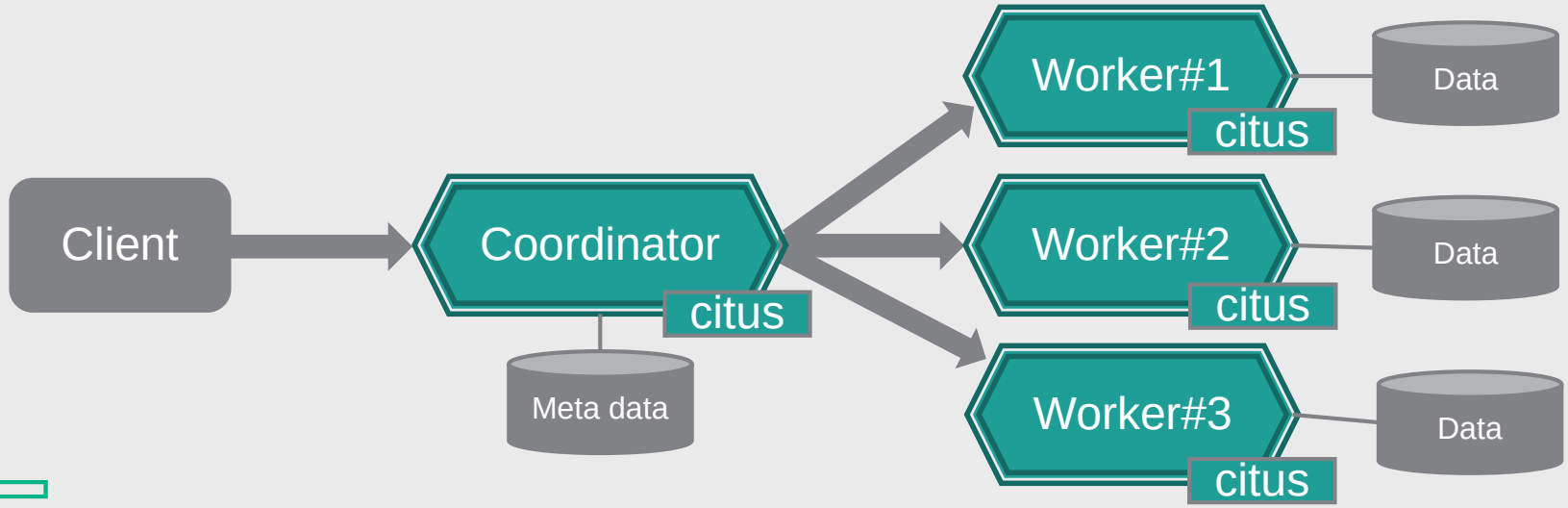
✓Coordinator Node

- PostgreSQL instance that accepts connections from client
- Manage meta-data

✓Worker Node

- PostgreSQL instance that manages the actual data
- Do not communicate between Worker Nodes

✓Install citus EXTENSION on all nodes



What’s Citus?

citus EXTENSION

✓Installation

- Install on Coordinator Node and Worker Node
- Execute the CREATE EXTENSION statement on all databases for creating tables
- Installation binaries are the same on all nodes
- Install extensions required for the application (such as pgcrypto) also to all nodes

What's Citus?

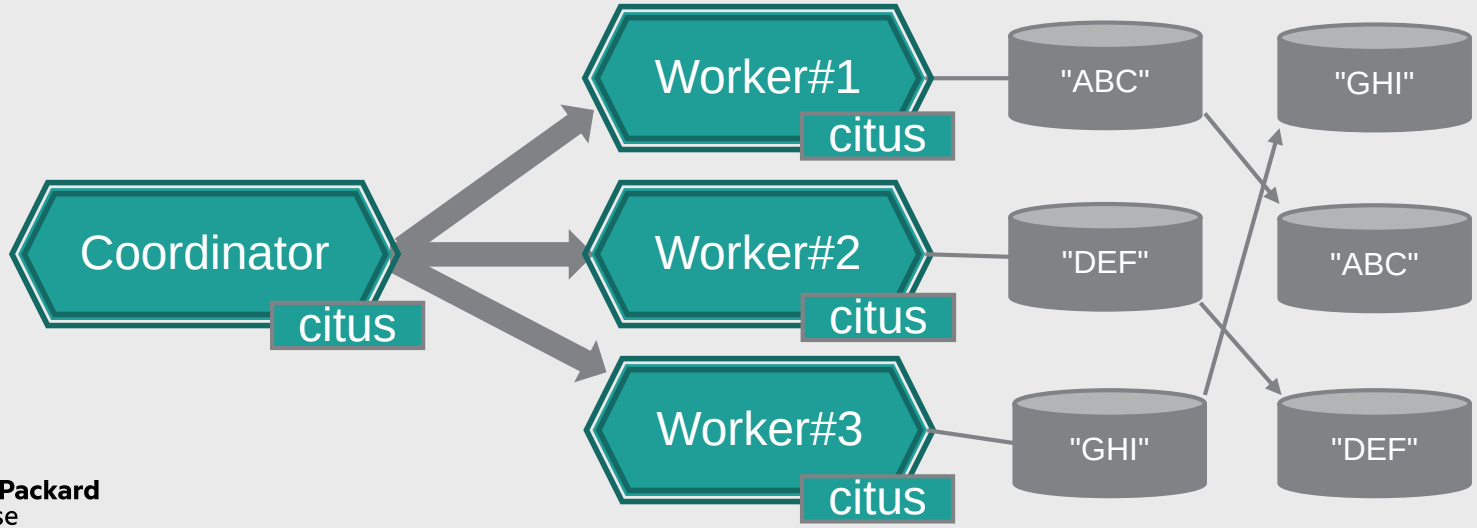
Table configuration

✓Distributed Table

- Table that stores the data distributedly
- Suitable for fact table
- Specify a column as distribution key (determined distribution destination table by the scope of hash values)
- Specify the number of partitions

  Parameter citus.shard_count (default 32)
- Can create replicas on difference Worker Nodes

  Parameter citus.shard_replication_factor (default 1 = No replica provided)

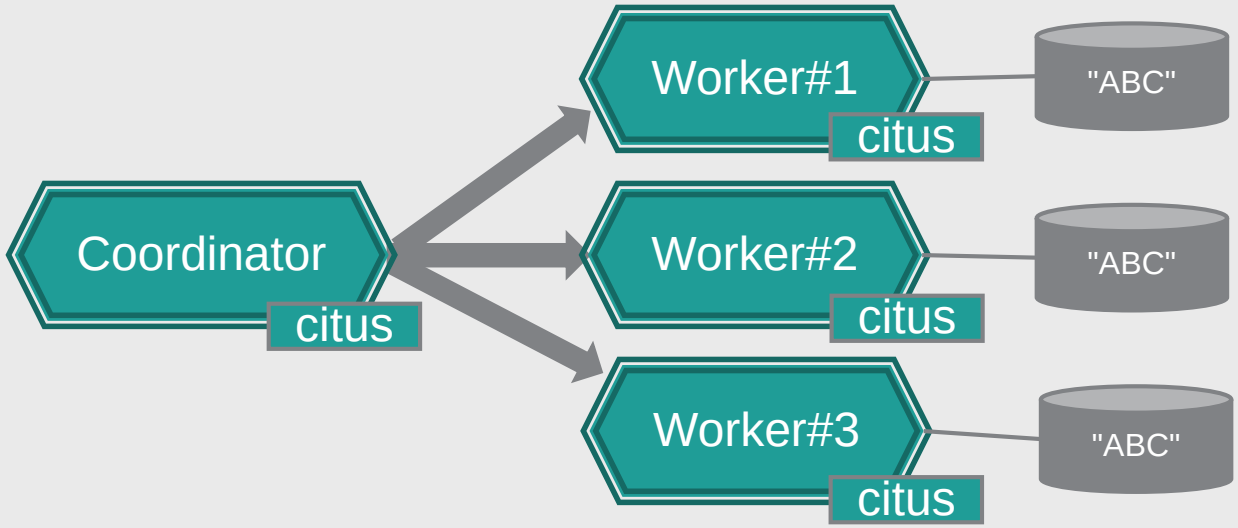


What's Citus?

Table configuration

✓Reference Table

- Table that stores the same data in all nodes
- Suitable for dimension table



Let’s try!

Let’s try!

Install

✓Build from source code

```
$ ./configure
$ make
# make install
```

✓Confirmation of EXTENSION

```
postgres=# SELECT * FROM pg_available_extensions WHERE
        name='citus' ;

-[ RECORD 1 ]-----+---------------------------
name              | citus
default_version   | 8.0-8
Installed_version |
comment           | Citus distributed database
```

Let’s try!

Operation on all instances

✓Specify 'citus' for parameter shared_preload_libraries

```
postgres=# SHOW shared_preload_libraries ;
 shared_preload_libraries
--------------------------
 citus
(1 row)
```

✓Loading EXTENSION

```
postgres=# CREATE EXTENSION citus ;
CREATE EXTENSION
```

Let’s try!

Authentication settings

- ✓Libpq connection from Coordinator Node to Worker Node
- ✓There is no password authentication mechanism for communication between Coordinator Node and Worker Node
- ✓Password less connection setup required
- ✓Configuration example of pg_hba.conf file (Worker Node)

```
host    all     all     coordhost1/32   trust
```

- ✓Also possible to use .pgpass file (Coordinator Node)
- ✓Authentication information can be specified in the parameter citus.node_conninfo (SSL setting etc)

Let’s try!

Operation on Coordinator Node

✓Register Worker Node to Coordinator Node (for each database)

✓Specify host name and port number in master_add_node function

```
postgres=# SELECT master_add_node('wrkhost1', 5432) ;
master_add_node
----------------------------------------------
 (1,1,wrkhost1,5432,default,f,t,primary,default)
(1 row)
```

Let’s try!

Operation on Coordinator Node

✓Confirmation

```
postgres=> SELECT nodeid, nodename, isactive FROM pg_dist_node ;
 nodeid | nodename  | isactive
--------+-----------+----------
      1 | wrkhost1  | t
      2 | wrkhost2  | t
      3 | wrkhost3  | t
(3 rows)
```

Let’s try!

Create Distributed Table

✓Specifying the number of distributed tables and the number of replicas

```
postgres=> SET citus.shard_count = 6 ;
SET
postgres=> SET citus.shard_replication_factor = 2 ;
SET
```

✓Example for table creation

```
postgres=> CREATE TABLE dist1(key1 NUMERIC, val1 VARCHAR(10)) ;
CREATE TABLE
postgres=> SELECT create_distributed_table('dist1', 'key1') ;
 create_distributed_table
--------------------------


(1 row)
```

Let’s try!

Create Distributed Table

✓Same configuration of the table is automatically created in the Worker Node

- The table name is "{Origin table name}_{ShardID}"
- TABLESPACE clause does not propagate

✓Number of tables created on Worker Node

- In the example of the previous slide, the distributed table# 6 x replica# 2 / Worker Node# 3 = 4 table is created

Let’s try!

Create Distributed Table

✓Due to replica settings, tables of the same name are created in different Worker Nodes.

- The same data is stored in the same name table

Coordinator	Worker#1	Worker#2	Worker#3
dist1	dist1_102046	dist1_102046	
		dist1_102047	dist1_102047
	dist1_102048		dist1_102048
	dist1_102049	dist1_102049	
		dist1_102050	dist1_102050
	dist1_102051		dist1_102051

Let’s try!

Create Reference Table

✓Table creation

```
postgres=> CREATE TABLE ref1(key1 NUMERIC, val1 VARCHAR(10)) ;
CREATE TABLE
postgres=> SELECT create_reference_table('ref1') ;
 create_reference_table
------------------------


(1 row)
```

✓Table confirmation on Worker Node

```
postgres=> ¥d
           List of relations
 Schema |      Name      | Type  | Owner
--------+----------------+-------+-------
 public | ref1_102084    | table | demo
(1 row)
```

Architecture

Architecture

Processes

✓bgworker: task tracker

- Running on Coordinator Node and Worker Node

✓bgworker: Citus Maintenance Daemon

- Running on Coordinator Node and Worker Nod
- Send usage data to https://reports.citusdata.com every 24 hours

✓Backend processes

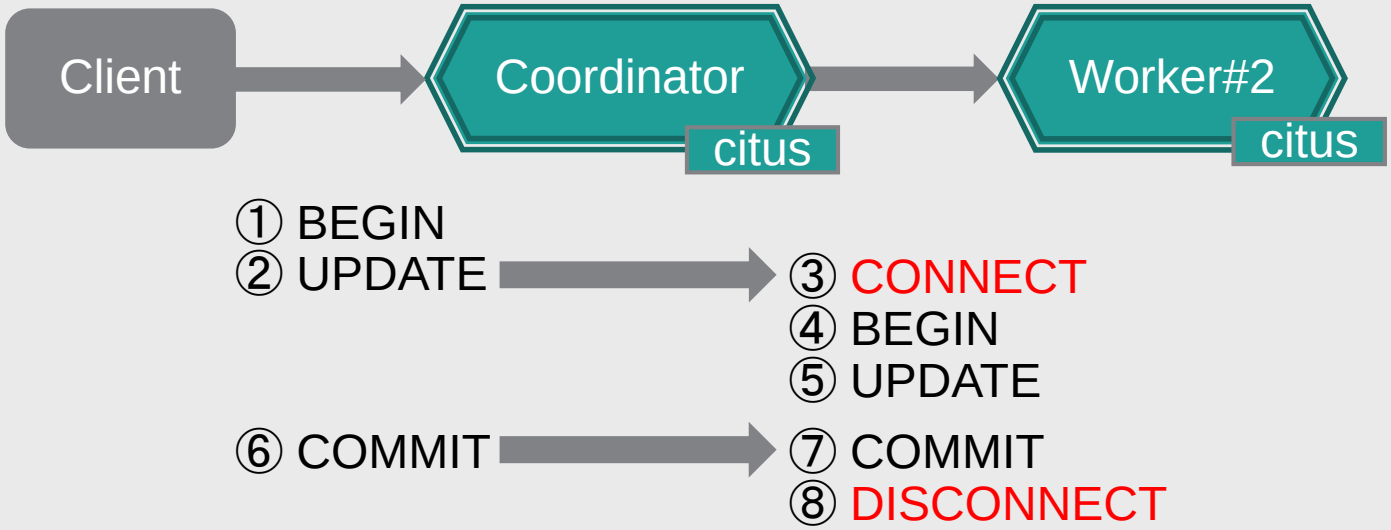
- Start up on Worker Node by connecting from Coordinator Node
- Execute SQL statement to Distributed Table or Reference Table
- Execute dump_local_wait_edges function every 2 seconds
- Search pg_prepared_xacts view every minute

Architecture

Session management

✓Connection between Coordinator Node and Worker Node

- Establishing session when executing the first SQL statement in the transaction
- Disconnect when transaction is completed
- Connection pool is not used (Available on Enterprise Edition)



Architecture

Executed SQL

✓Process to be executed on Coordinator Node

- Final sort (ORDER BY)
- Operation of SEQUENCE (Include SERIAL column and GENERATED AS IDENTITY column)
- Processing objects other than tables and indexes

✓Process to propagate to Worker Node

- VACUUM statement
- ANALYZE statement
- CREATE INDEX statement
- ALTER TABLE statement (some restrictions)

✓Other DML

- Select the table on the Worker Node that issues the SQL statement by specifying the distributed key column

Architecture

Executed SQL

✓PostgreSQL API to submit SQL to Worker Node

- PQsendQuery
- PQsendQueryParams
- Use asynchronous API

Architecture

Executed SQL

✓When the distributed key string can be specified

✓SQL executed by the application ⇒ SQL executed on Worker Node

```
SELECT * FROM dist1 WHERE key1 = 20
```

```
SELECT key1, val1 FROM public.dist1_102131 dist1 WHERE
        (key1 OPERATOR(pg_catalog.=) 20)
```

```
UPDATE dist1 SET val1 = 'update' WHERE key1 = 20
```

```
UPDATE public.dist1_102131 dist1 SET val1 = 'update'::character
varying WHERE (key1 OPERATOR(pg_catalog.=) 20)
```

Architecture

Executed SQL

✓Access to specific Worker Node only when Distributed Key can be specified

```
postgres=> EXPLAIN SELECT * FROM dist1 WHERE key1 = 1000 ;
                          QUERY PLAN
-----------------------------------------------------------------
 Custom Scan (Citus Router)  (cost=0.00..0.00 rows=0 width=0)
   Task Count: 1
   Tasks Shown: All
   ->  Task
         Node: host=wrkhost1 port=5432 dbname=postgres
         ->  Seq Scan on dist1_102078 dist1  (cost=0.00..2973.04
rows=1 width=12)
               Filter: (key1 = '1000'::numeric)
(7 rows)
```

Architecture

Executed SQL

✓When the distributed key can not be specified

✓SQL executed by the application ⇒ SQL executed on Worker Node

```
SELECT * FROM dist1 WHERE key1 != 20
```

```
COPY (SELECT key1, val1 FROM dist1_102146 dist1 WHERE
        (key1 OPERATOR(pg_catalog.<>) 20)) TO STDOUT

COPY (SELECT key1, val1 FROM dist1_102147 dist1 WHERE
        (key1 OPERATOR(pg_catalog.<>) 20)) TO STDOUT
...
```

- Changing the table name and putting it to all Worker Nodes

Architecture

Executed SQL

✓Execution plan when the distributed key can not be specified

```
postgres=> SET citus.explain_all_tasks = on ;
SET
postgres=> EXPLAIN SELECT * FROM dist1 ;
                          QUERY PLAN
------------------------------------------------------------------
Aggregate  (cost=0.00..0.00 rows=0 width=0)
   ->  Custom Scan (Citus Real-Time)  (cost=0.00..0.00 rows=0
width=0)
         Task Count: 6
         Tasks Shown: All
         ->  Task
               Node: host=wrkhost1 port=5001 dbname=demodb
               ->  Aggregate  (cost=2973.04..2973.05 rows=1 width=8)
...
```

Architecture

Executed SQL

✓Joining Distributed Table and Reference Table

- Joining in Worker Node

✓SQL executed by the application ⇒ SQL executed on Worker Node

```
SELECT * FROM dist1 d1 INNER JOIN ref1 r1 ON d1.key1=r1.key1 WHERE
d1.key1=2
```

⇓

```
SELECT d1.key1, d1.val1, r1.key1, r1.val1 FROM (public.dist1_102221
d1 JOIN public.ref1_102084 r1 ON ((d1.key1 OPERATOR(pg_catalog.=)
r1.key1))) WHERE (d1.key1 OPERATOR(pg_catalog.=) (2)::numeric)
```

Architecture

Executed SQL

✓Joining Distributed Tables

- Error may occur by default
- Executable by specifying the parameter citus.enable_repartition_joins to 'on'

✓Recommends placing the table on the same node with the same column value

```
CREATE TABLE event(tenant_id INT, event_id BIGINT, … ) ;
SELECT create_distributed_table('event', 'tenant_id') ;

CREATE TABLE page(tenant_id INT, page_id INT, … ) ;
SELECT create_distributed_table('page', 'tenant_id',
        colocate_with => 'event') ;
```

Architecture

Parameter settings

✓Major parameters (38 in total)

- citus.node_connection_timeout
- citus.partition_buffer_size
- citus.recover_2pc_interval
- citus.remote_task_check_interval
- citus.shard_count
- citus.shard_max_size
- citus.shard_placement_policy
- citus.shard_replication_factor
- citus.subquery_pushdown
- citus.task_assignment_policy
- citus.task_executor_type
- citus.task_tracker_delay
- citus.use_secondary_nodes
- …

Architecture

Catalogs

✓Major catalogs (11 in total)

Catalog name	Description
pg_dist_authinfo	Store connection information (Enterprise Edition)
pg_dist_colocation	Co-location groups is stored
pg_dist_node	Information about the worker nodes
pg_dist_node_metadata	Information about ServerID
pg_dist_partition	Information about the distribution column
pg_dist_placement	State of shard placements
pg_dist_poolinfo	Information about Connection pooling (Enerprise Edition)
pg_dist_shard	Information about distributed table

Restriction

Restriction

SQL can not be executed

✓The following syntax can not be executed for Distributed Table

- Updating distributed key columns (UPDATE / INSERT ON CONFLICT)
- SELECT FOR UPDATE statement (if creating a replica)
- TABLESAMPLE clause
- WITH RECURSIVE clause
- Generate_series function for INSERT VALUES statement

✓Described in the manual

```
postgres=> BEGIN ;
BEGIN
postgres=> SELECT * FROM dist1 WHERE key1=100 FOR UPDATE ;
ERROR:  could not run distributed query with FOR UPDATE/SHARE
commands
HINT:  Consider using an equality filter on the distributed table's
partition column.
```

Restriction

Restricted SQL

✓The following syntax is restricted for execution

- Correlated subquery
- GROUPING SETS clause
- PARTITION BY clause
- Joining Local Table and Distributed Table
- Trigger created on Coordinator Node
- INSERT SELECT ON CONFLICT statement

```
postgres=> SELECT COUNT(*) FROM dist1 d INNER JOIN local1 l ON
        d.key1 = l.key1 ;
ERROR:  relation local1 is not distributed

postgres=> SELECT COUNT(*) FROM dist1 d INNER JOIN
        (SELECT * FROM local1) l ON d.key1 = l.key1 ;
```

Restriction

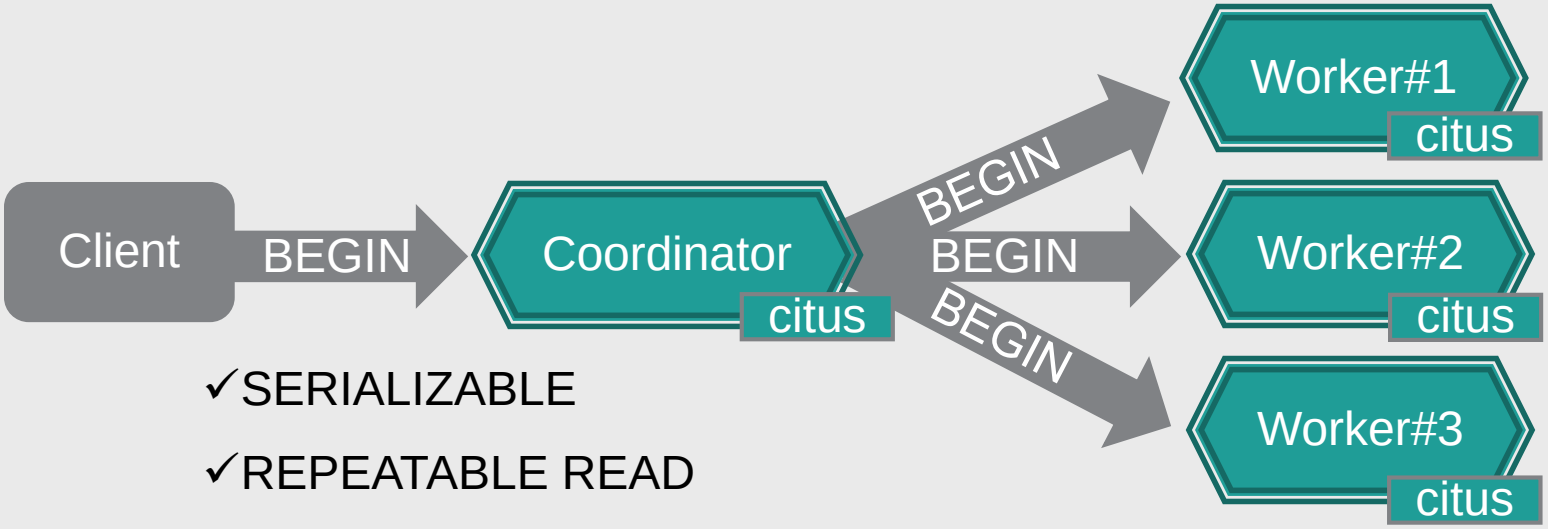
ISOLATION LEVEL

✓Connection from client to Coordinator Node

- No restriction

✓Connection from Coordinator Node to Worker Node

- READ COMMITED (hard-coded)



✓SERIALIZABLE

✓REPEATABLE READ

✓READ COMMITED

✓READ UNCOMMITED

✓READ COMMITED

Restriction

ALTER TABLE statement

✓ALTER TABLE statement can only be executed as follows

- Add / Drop columns
- Setting restriction
- Partition administration
- Modify data type of columns

✓Control automatic propagation of DDL

- Parameter citus.enable_ddl_propagation (default 'on')

```
postgres=> ALTER TABLE dist1 SET UNLOGGED ;
ERROR:  alter table command is currently unsupported
DETAIL:  Only ADD|DROP COLUMN, SET|DROP NOT NULL, SET|DROP DEFAULT,
ADD|DROP CONSTRAINT, SET (), RESET (), ATTACH|DETACH PARTITION and
TYPE subcommands are supported.
```

Restriction

SQL that does not propagate

✓DATABASE and USER information should be identical in all nodes

- CREATE USER / CREATE DATABASE statement does not propagate
- Warning is output
- Function run_command_on_workers is provided to execute SQL statements on Worker Node

```
postgres=# CREATE DATABASE demodb ;
NOTICE:  Citus partially supports CREATE DATABASE for distributed
databases
DETAIL:  Citus does not propagate CREATE DATABASE command to workers
HINT:  You can manually create a database and its extensions on
workers.
CREATE DATABASE
```

Behavior when trouble occurs

Behavior when trouble occurs

Coordinator Node down

- ✓Client can not connect if Coordinator Node down
- ✓Automatic failover feature is not provided
- ✓Streaming Replication + Clusterware are required

Behavior when trouble occurs

Worker Node down

✓When the Worker Node is down, SQL that updates the entire data including the stopped node can not be executed

```
postgres=> DELETE FROM dist1 ;
ERROR:  connection error: wrkhost1:5432
DETAIL:  could not connect to server: Connection refused
        Is the server running on host "workhost1" (192.168.1.101)
and accepting
        TCP/IP connections on port 5432?
```

✓SQL which operates other than the data of the stopped node is warned but can be executed

```
postgres=> SELECT COUNT(*) FROM dist1 WHERE key1 = 100 ;
WARNING:  connection error: wrkhost1:5432
DETAIL:  could not send data to server: Connection refused
Could not send startup packet: Connection refused
...
```

Behavior when trouble occurs

Worker Node down

✓Warnings are output for records with replicas, but updatable

- Maintenance of replica of stopped Worker Node is not maintained

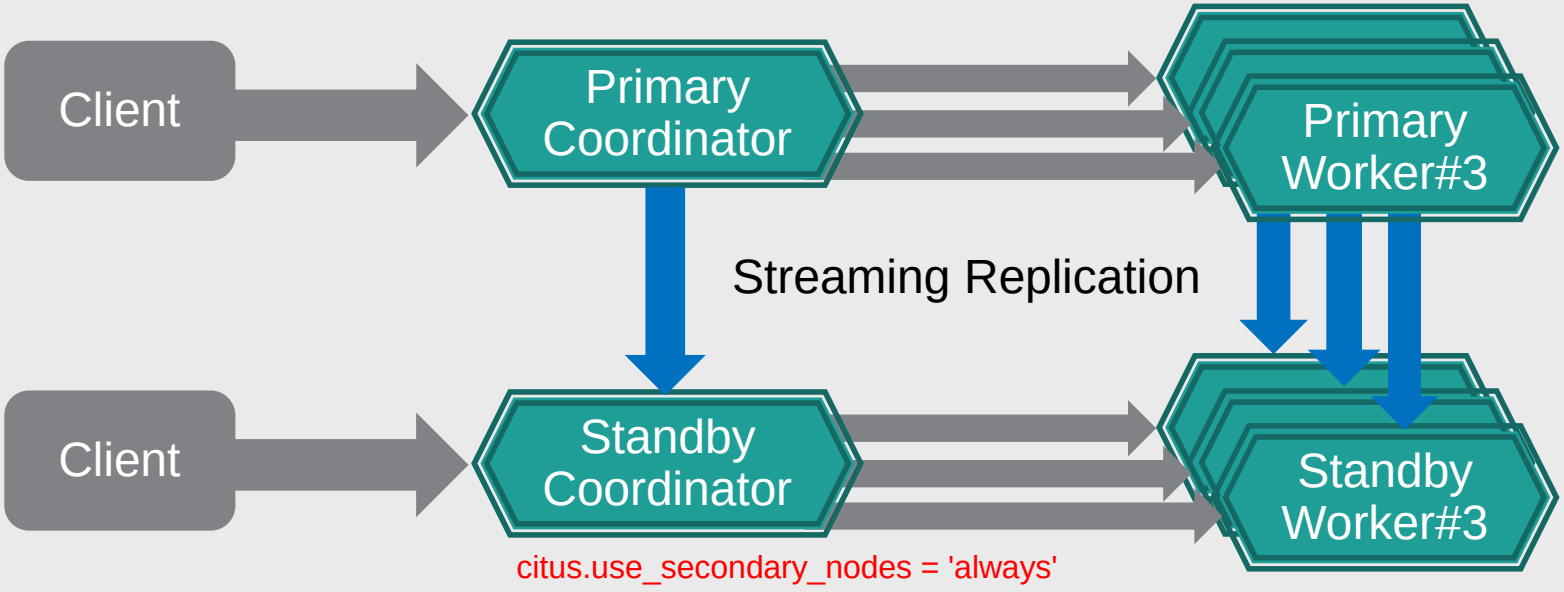
```
postgres=> DELETE FROM dist1 WHERE key1 = 1 ;
WARNING:  connection error: wrkhost1:5432
DETAIL:  could not send data to server: Connection refused
could not send SSL negotiation packet: Connection refused
DELETE 1
```

Behavior when trouble occurs

Combined with streaming replication

✓Streaming replication environment for load balancing is available

- Create streaming replication environment for all nodes
- Set citus.use_secondary_nodes = 'always' for Standby instance of Coordinator Node
- SELECT statement is executed on the standby instance of the Worker Node
- Standby instance is registered in master_add_secondary_node function



Summary

Summary

Some restriction exists, but easy to scale-out

- ✓It is relatively easy to build a scale-out environment
- ✓Possibility of performance improvement by parallel query + partitioning across nodes
- ✓It is necessary to implement of fault tolerance and backup by yourself
- ✓Because restrictions of SQL statement exist, prior application verification is recommended
- ✓Please note the difference from Enterprise Edition

Summary

Reference information URL

- ✓Product manuals

  https://docs.citusdata.com/en/v8.0/

- ✓Performance comparison video

  https://www.youtube.com/watch?v=g3H4nGsJsl0

- ✓Getting Started

  https://docs.citusdata.com/en/v8.0/portals/getting_started.html

- ✓Use Cases

  https://docs.citusdata.com/en/v8.0/portals/use_cases.html

- ✓API / Reference

  https://docs.citusdata.com/en/v8.0/portals/reference.html

- ✓GitHub

  https://github.com/citusdata/citus

Hewlett Packard
Enterprise

Thank you

noriyoshi.shinoda@hpe.com