Postgres Conference

HangZhou, China

Postgres-XC/XL
Scale-out Approach in PostgreSQL

July 25th, 2015

NTT DATA INTELLILINK Corporation
Koichi Suzuki

NTT DATA

Introduction

About the Speaker

- Fellow at NTT DATA Intellilink Corporation
- Principal, Technology Professionals at NTT DATA Group

In Charge Of

- General Database Technology
- Database in huge data warehouse and its design
- PostgreSQL and its cluster technology

In The Past

- Character Set Standard (Extended Unix Code, Unicode, etc)
- Heisei-font development (Technical Committee)
- Oracle Porting
- Object-Relational Database

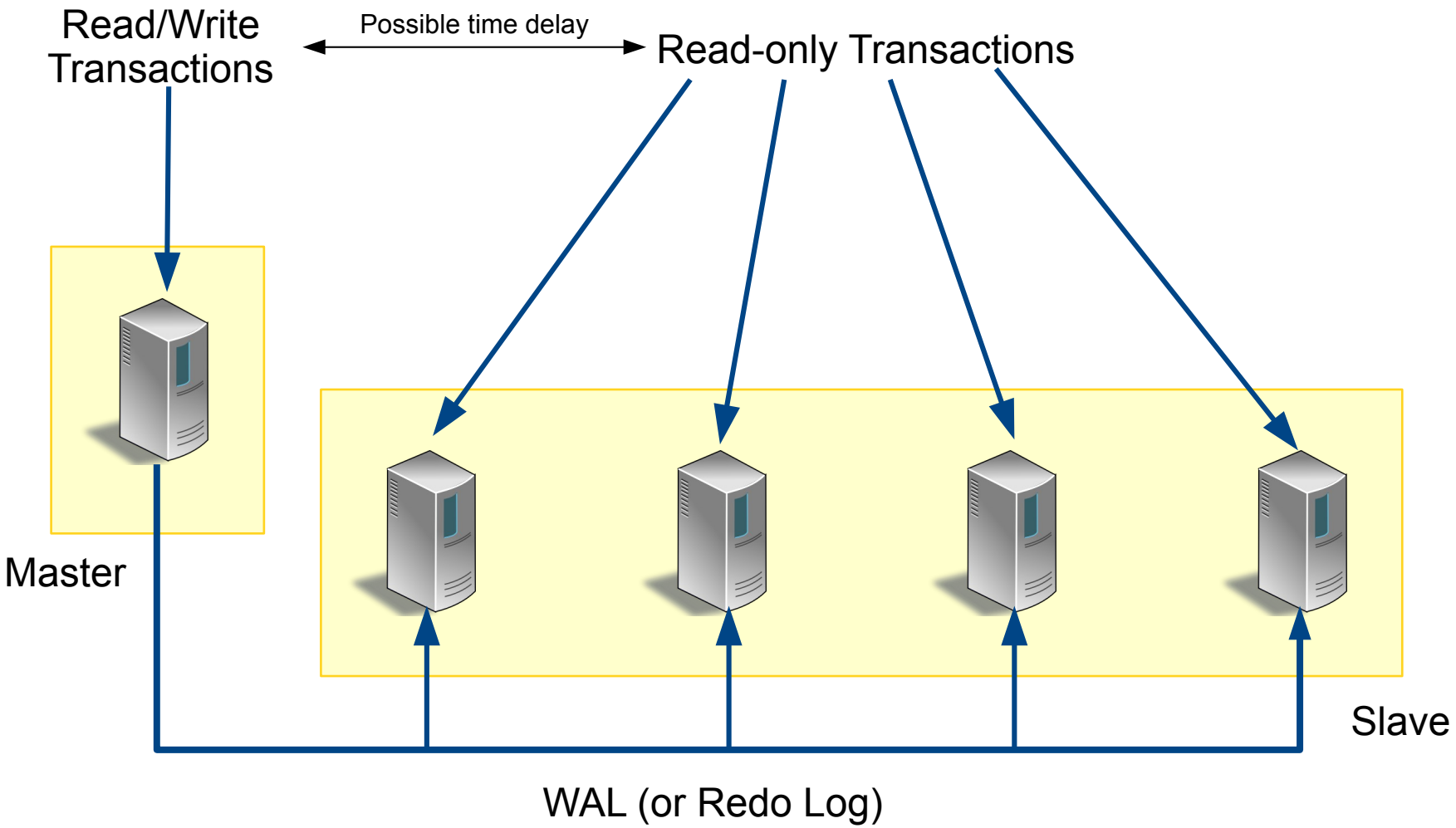
Motivation

- Growing Database Workload both in OLTP (OnLine Transaction Processing) and OLAP (OnLine Analytical Processing) applications.
- Shared-Nothing Approach
  - Performance with commodity hardware/software
- Extension to existing PostgreSQL
- Transparent API
  - Internal API could be different
  - Transparent libpq Interface
  - No significant restriction to transaction ACID properties and SQL language.

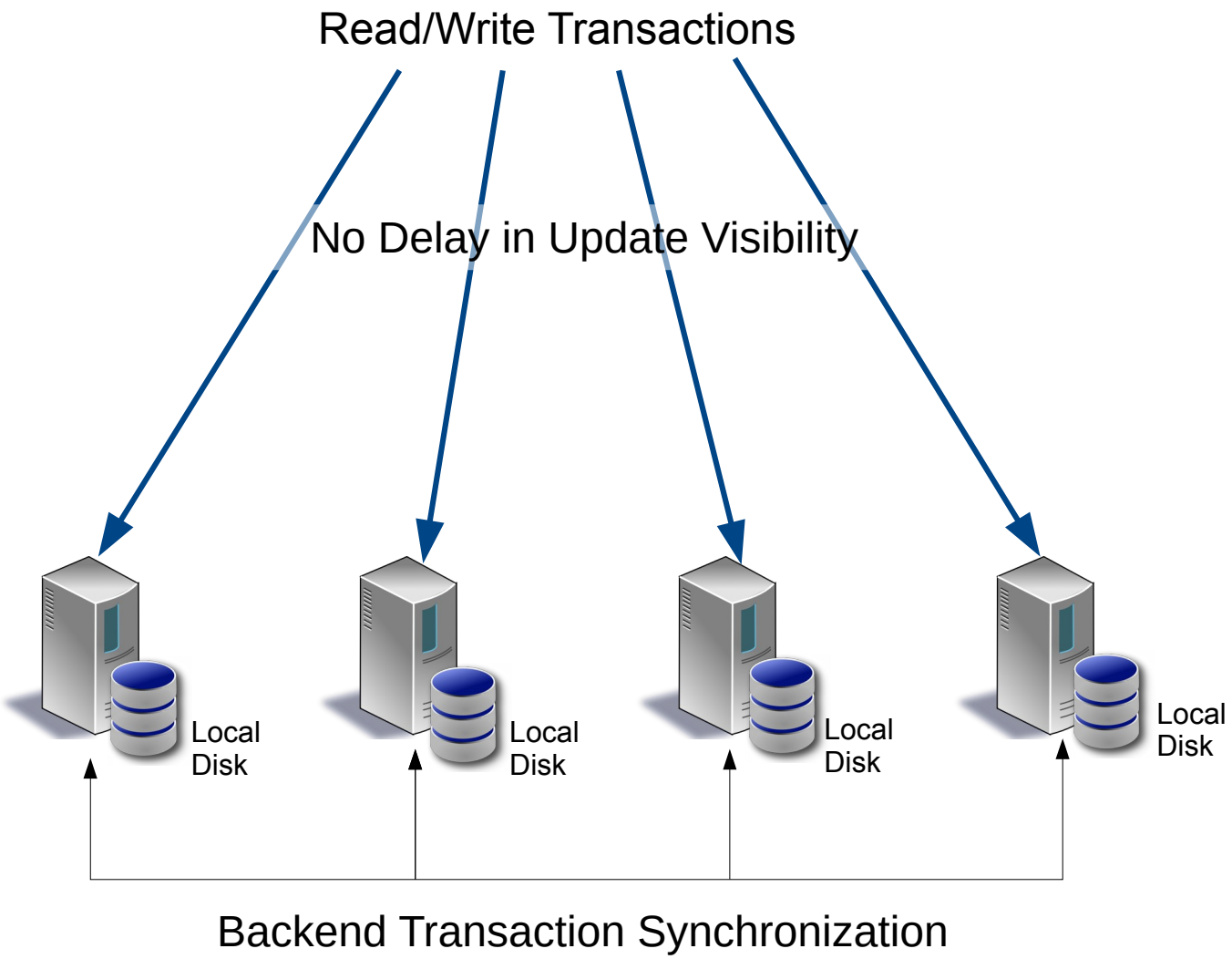
Scale-out approach

- Distribution/Replication of table rows among different database “nodes”
  - Parallelism
  - Local join operation
- SQL planning for row distribution/replication
- Consistent and synchronous transaction management among “nodes”
- Performance with commodity hardware/software

Read Scale-out in PostgreSQL Master/Slave

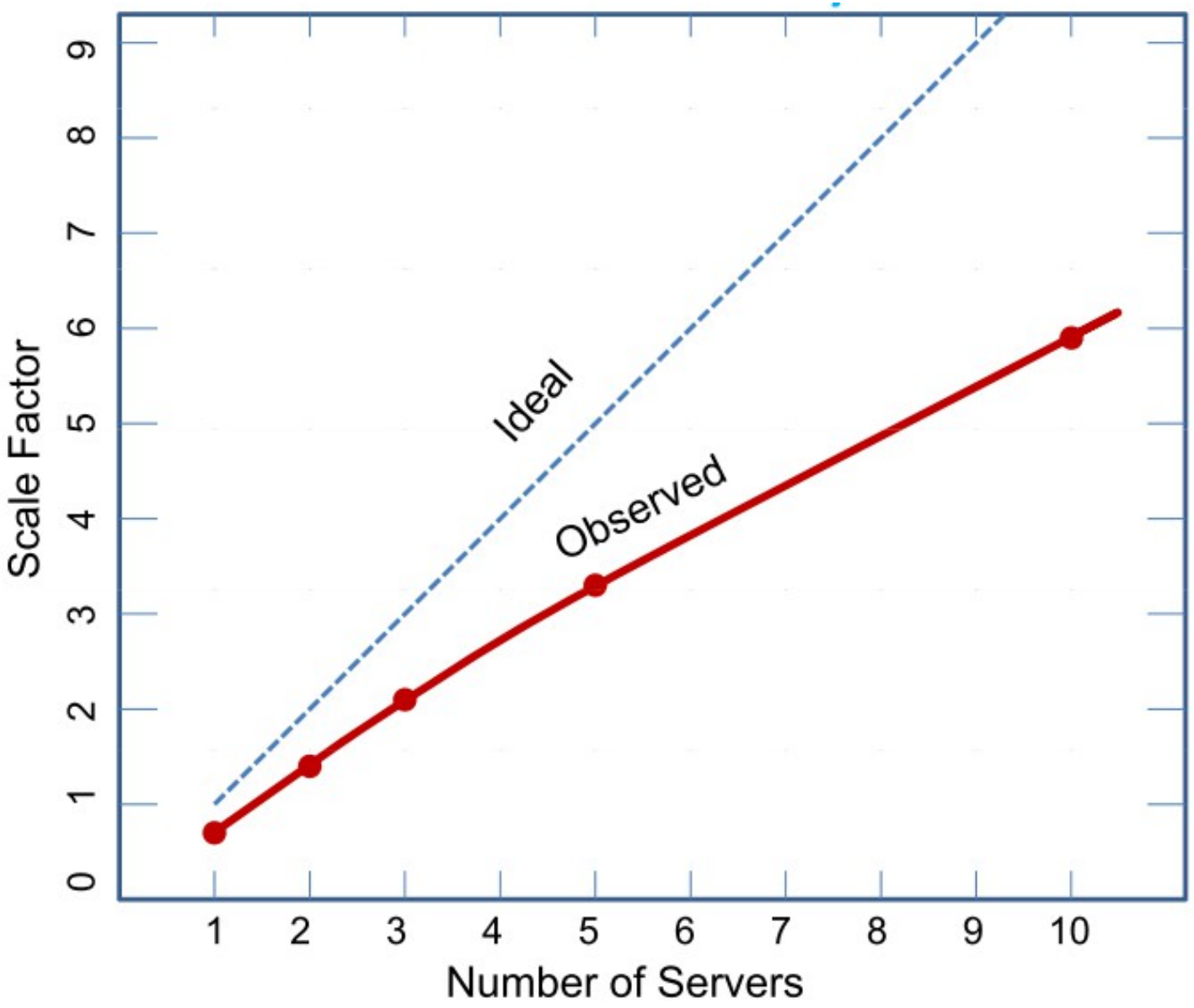


Scaling Out in Postgres XL/XC



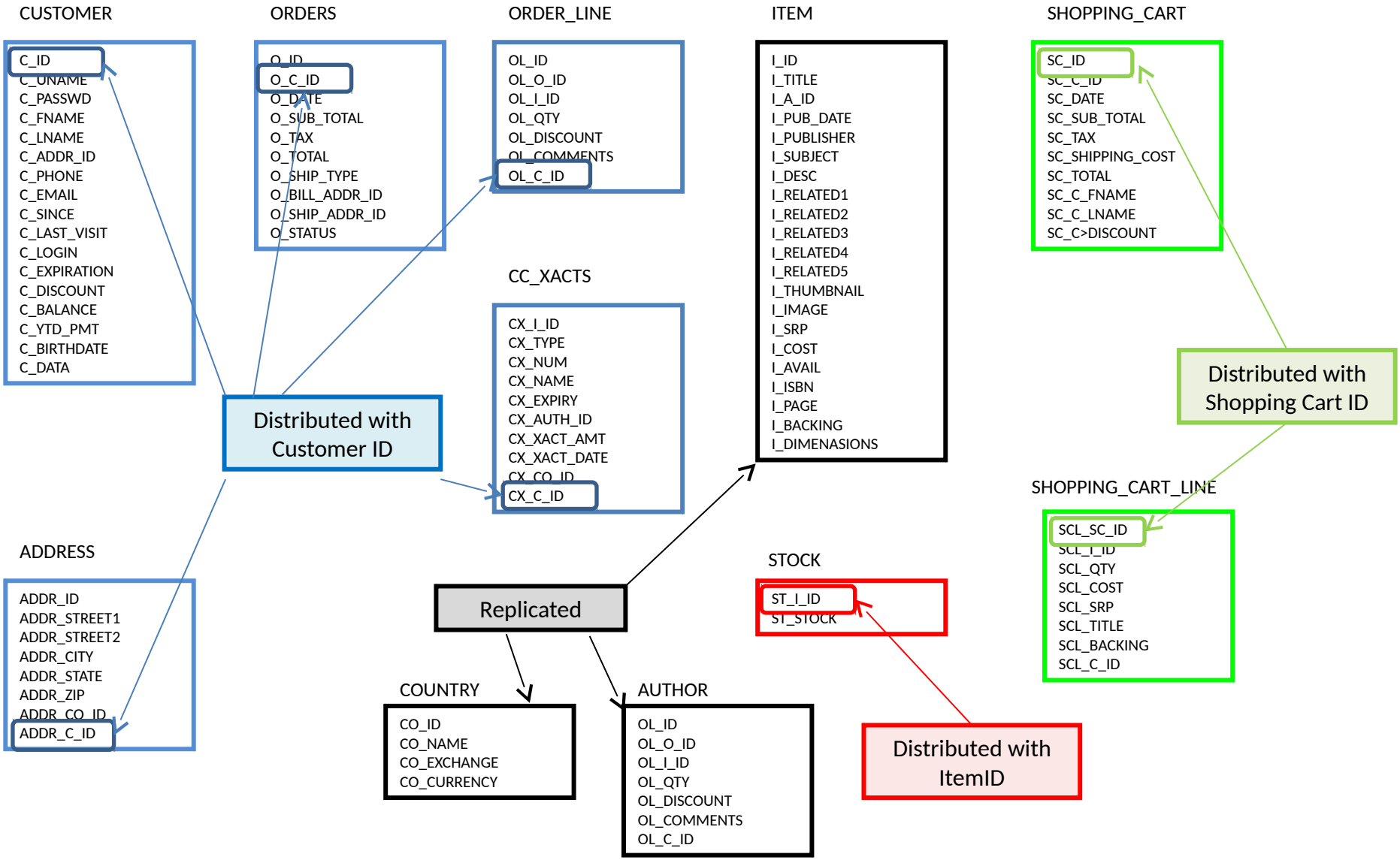
OLTP Workload Scalability and Table Design

DBT-1 Workload Scalability

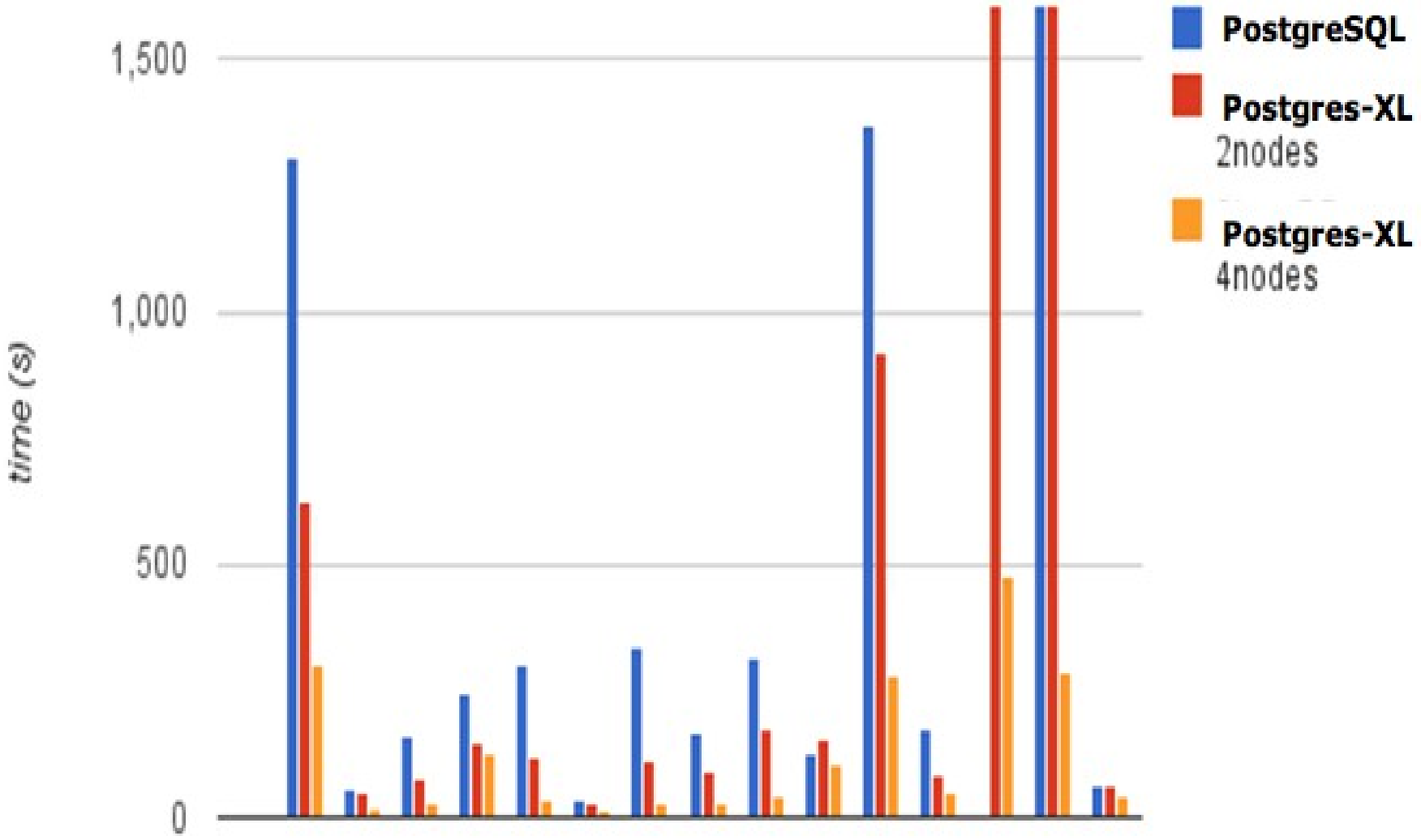


DBT-1 (Rev)

Table Design in DBT-1 Benchmark



MPP Performance – DBT-3 (TPC-H)



By courtesy of Mason Sharp, Postgres-XL leader

Scale Out Approach (1): Table Distribution/Replication

NTT DATA

Categorize tables into two groups:

Large and frequently-updated tables

→ Distribute rows among nodes (Distributed Tables)
→ Based on a column value (distribution key)
→ Hash, modulo or round-robin

→ Parallelism among transactions (OLTP) or in SQL processing (OLAP)

Smaller and stable tables

→ Replicate among nodes (Replicated Tables)

→ Join Pushdown

Avoid joins between Distributed Tables with join keys different from distribution key as possible.

Scale Out Table Design in DBT-1

Three distribution keys:

- Customer ID
- Shopping Cart ID
- Item ID

Some transactions involve joins across distributed tables with non-distribution join keys.

Some More in XL/XC Node Configuration

Node Configuration: Two-Tier Approach

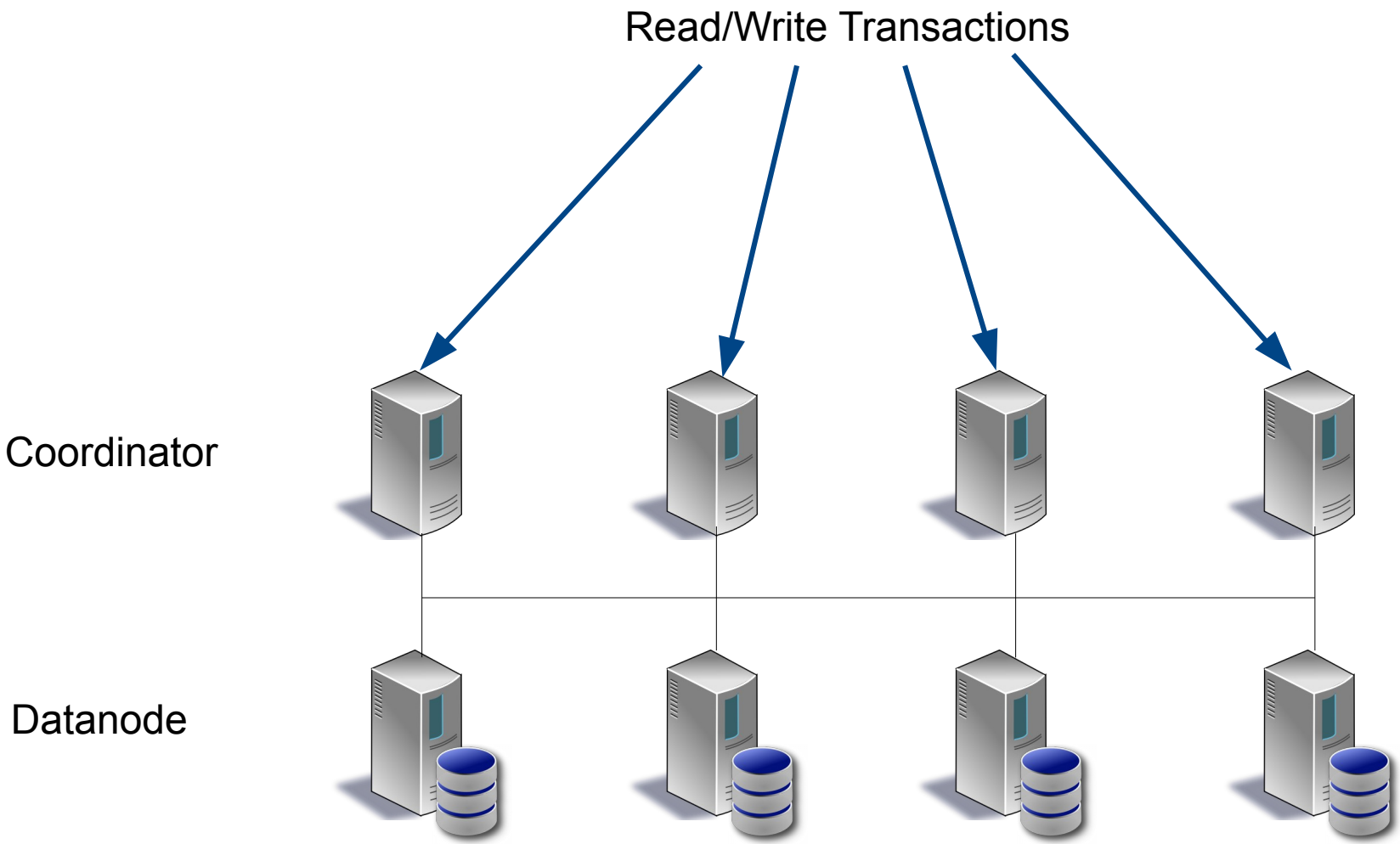
Coordinator:

- Maintains global catalog information
- Build global SQL plan and SQL statements for datanodes
- Interact with datanode to execute local SQL statements and accumulate the result

Datanode

- Maintains actual data (local data)
- Run local SQL statement from Coordinator
  (In XL, datanode may ask other datanodes for their local data)

Coordinator and Datanode



Node Configuration: Yet Another Node: GTM

GTM: Global Transaction Manager

Synchronizes each node's transaction status

Why GTM? Two-Phase Commit Protocol doesn't work?

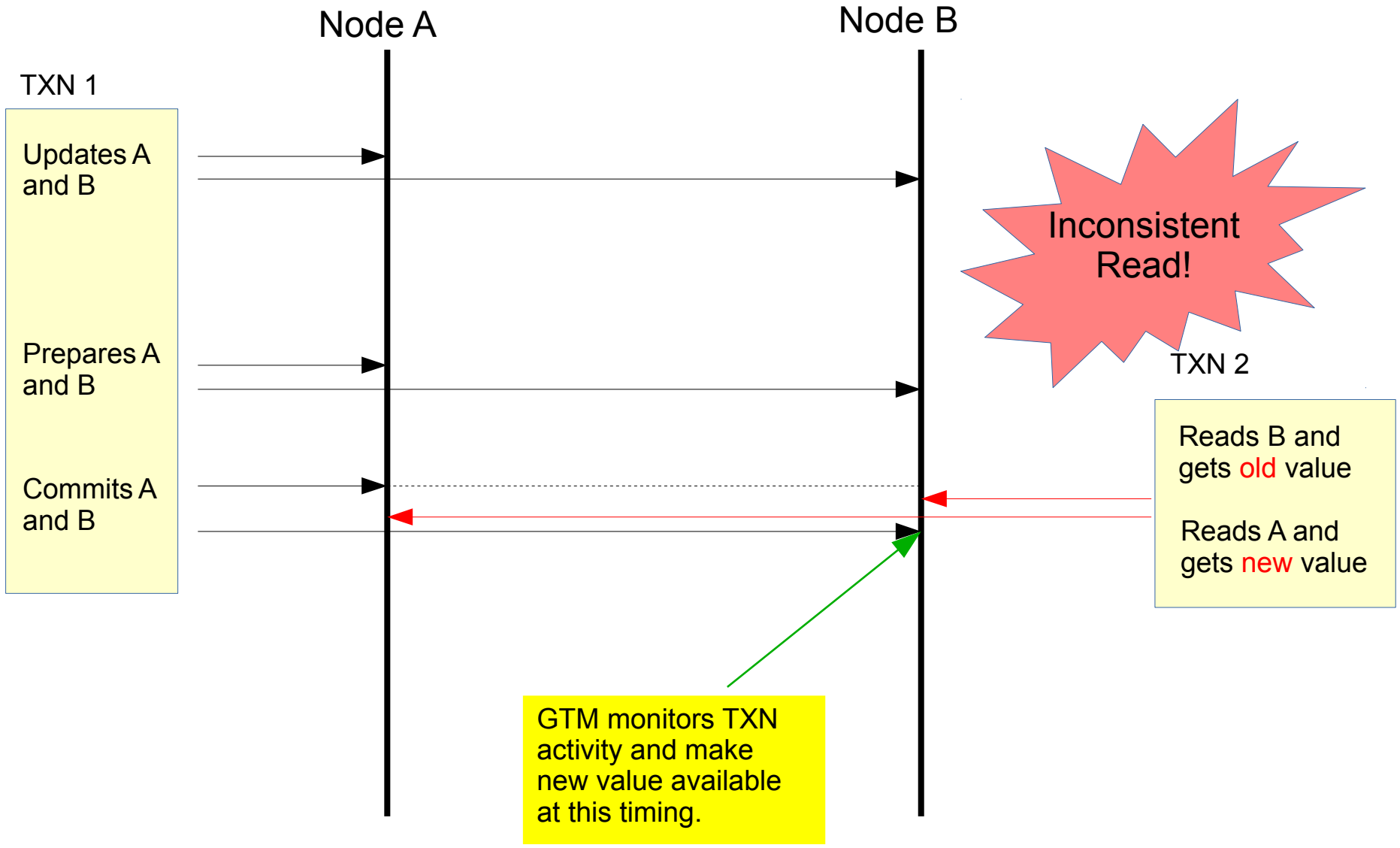
Two-Phase Commit Protocol Does:

- Maintain database consistency in transactions updating more than one node.

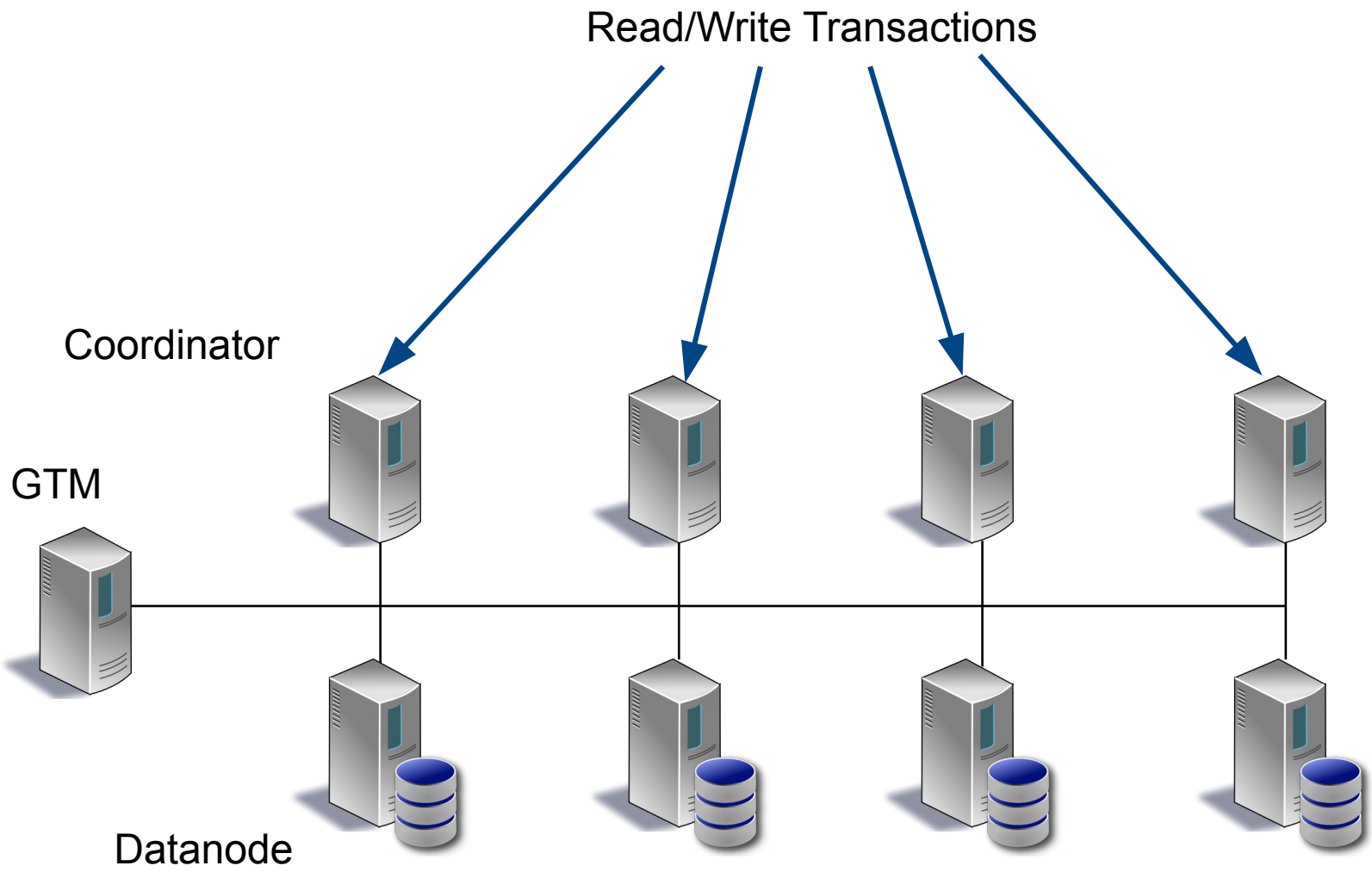
Two-Phase Commit Protocol Doesn't:

- Maintain Atomic Visibility of Updates to other transactions (next slide)

Atomic Visibility and GTM



Final Configuration: GTM, Coordinator and Datanode



Configuration in Practice

Just like configuring many database servers to talk each other

- Many pitfalls
- Pgxc_ctl provides simpler way to configure the whole cluster
  - Provide only needed parameters
  - Pgxc_ctl will do the rest to issue needed commands and SQL statements.

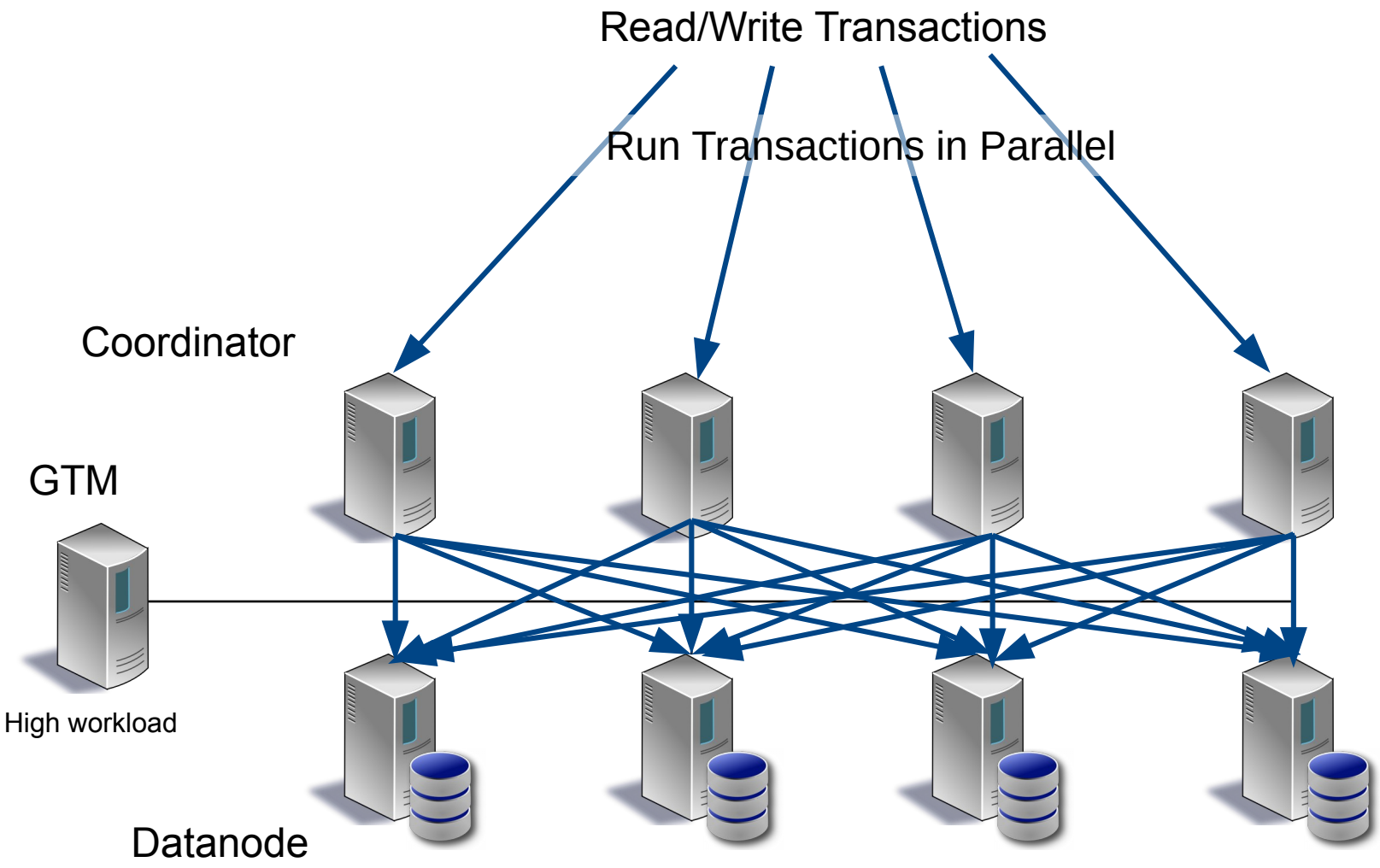
– Visit
`http://sourceforge.net/p/postgres-xc/xc-wiki/PGOpen2013_Postgres_Open_2013/`

Scalability in OLTP Workloads

OLTP Workload Characteristics

Number of Transactions: Many
Number of Involved Table Rows: Small
Locality of Row Allocation: High
Update Frequency: High

Scaling Out OLTP Workload

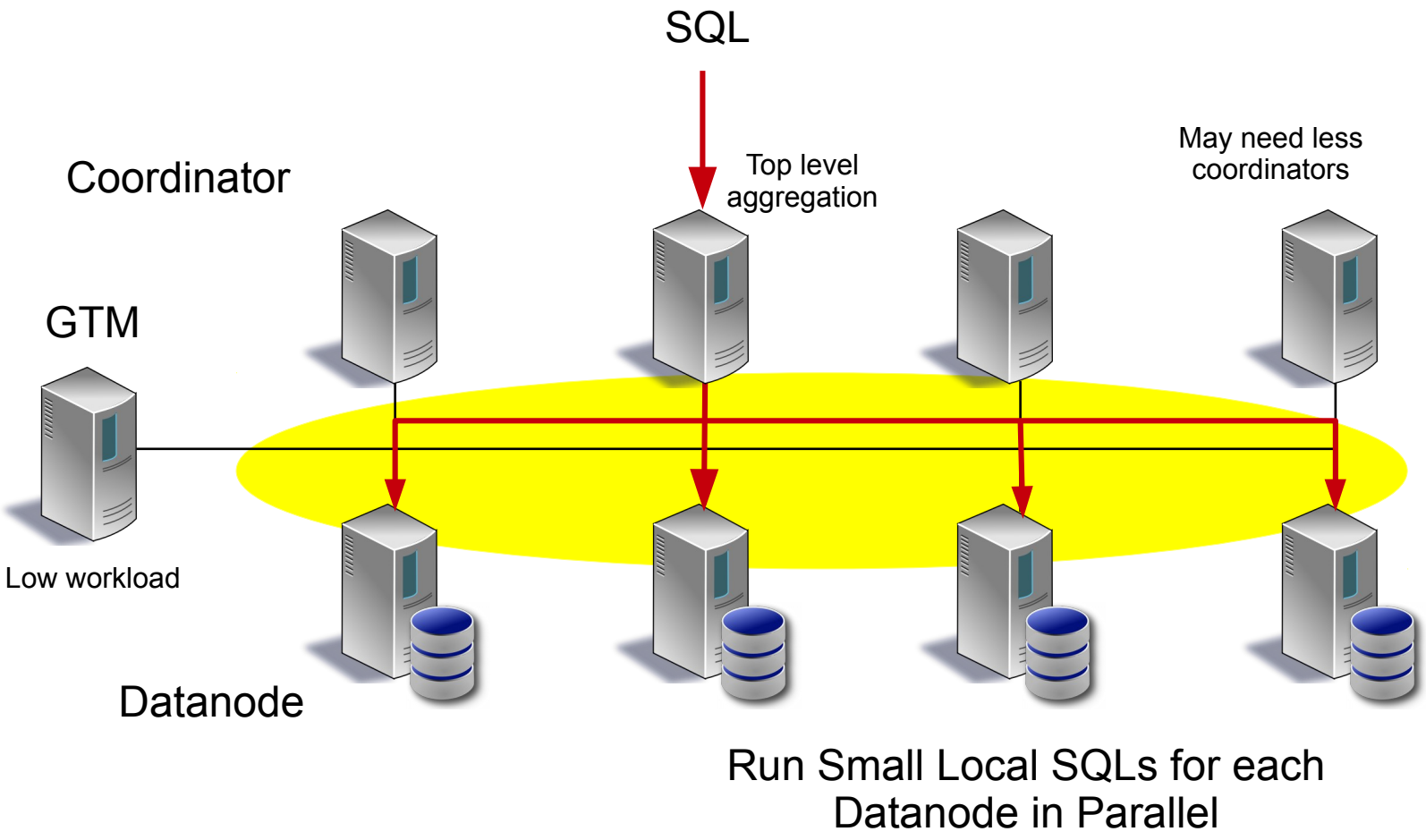


Scalability in OLAP (Analytic) Workloads

OLAP Workload Characteristics

Number of Transactions: Small
Number of Involved Table Rows: Huge
Locality of Row Allocation: Low
Update Frequency: Low

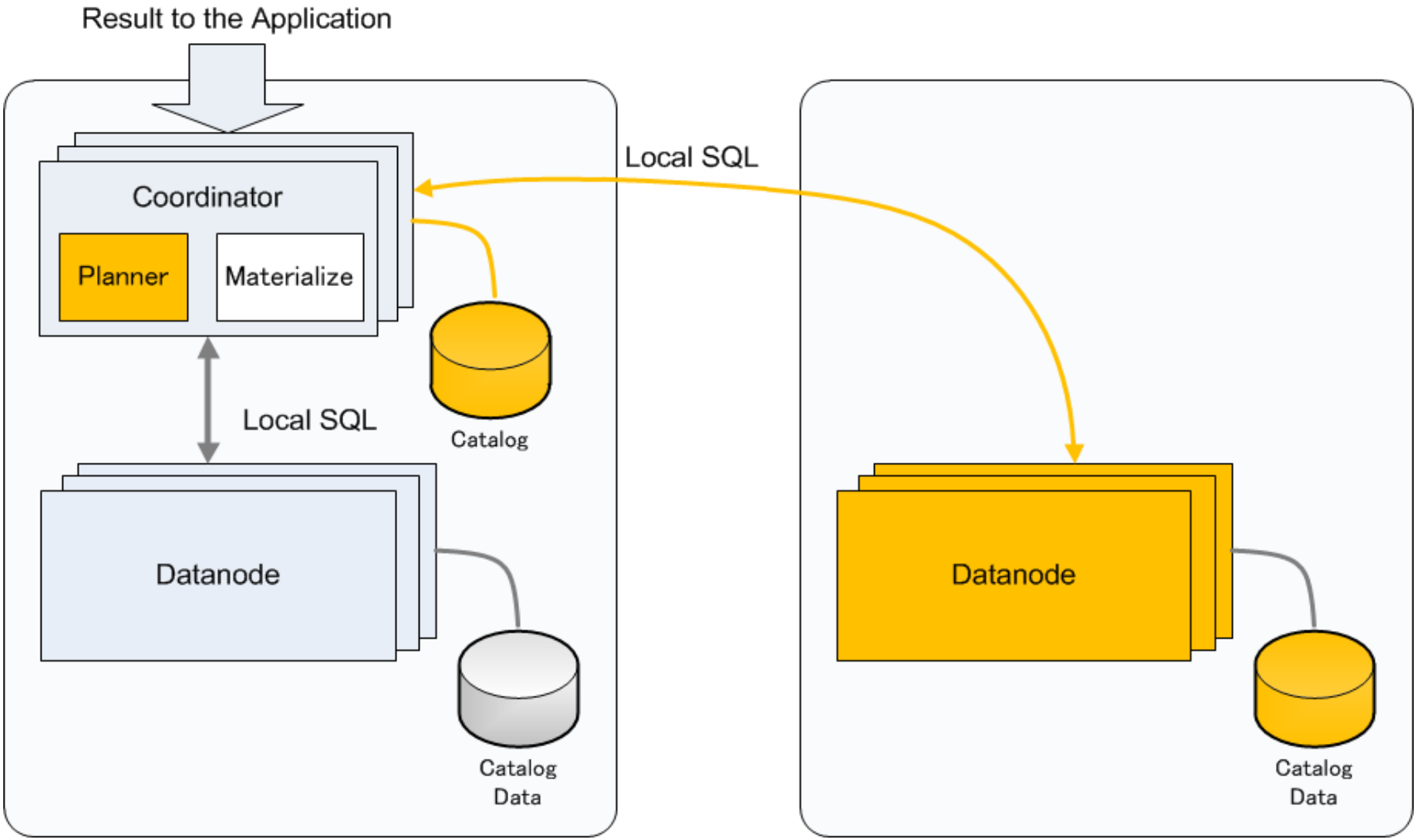
Scaling Out OLAP Workload



Join Offloading

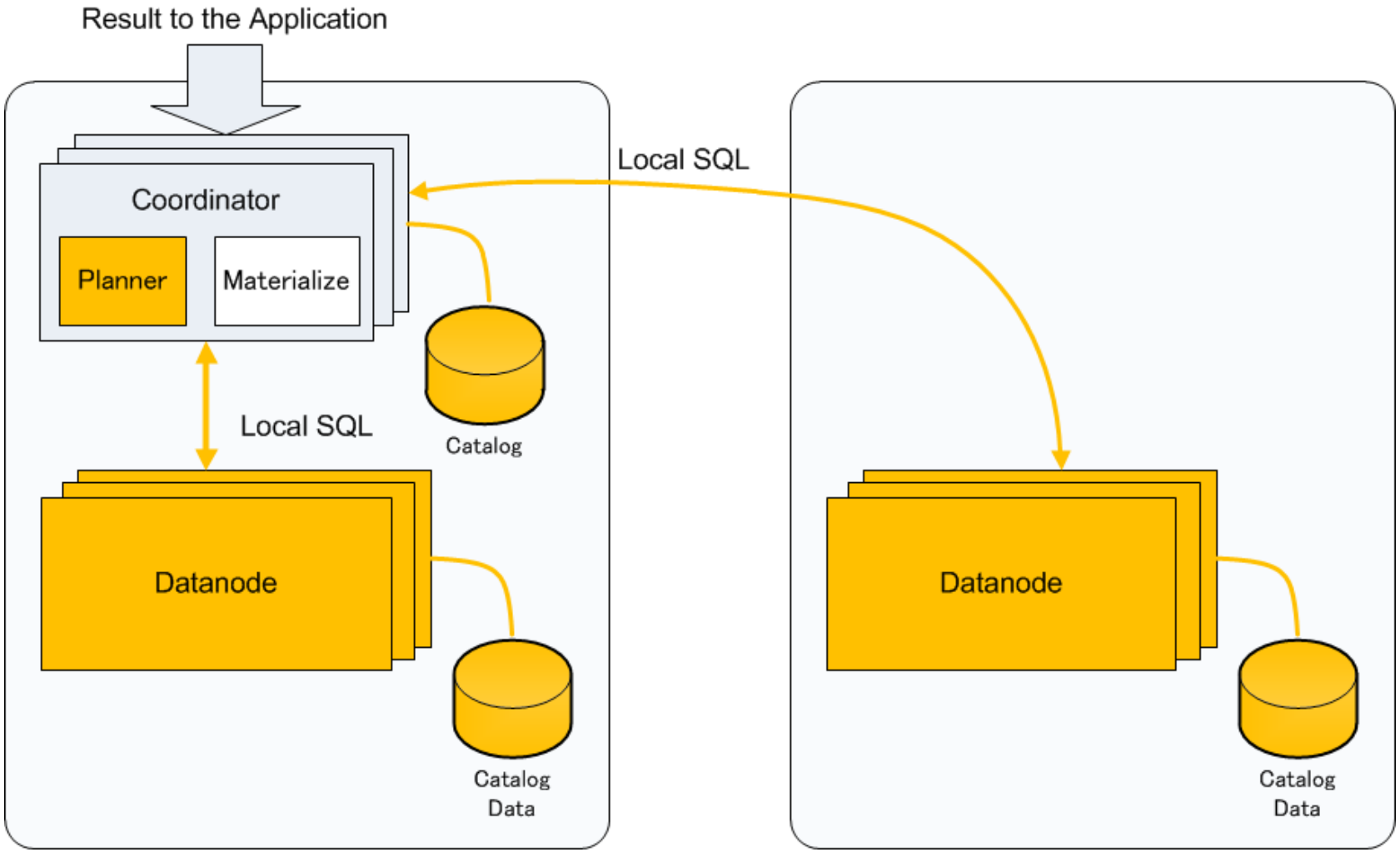
Join Offloading: When row allocation is available

- Replicated Table and Partitioned Table
  - Can determine which datanode to go from WHERE clause



Join Offloading: When row allocation is available

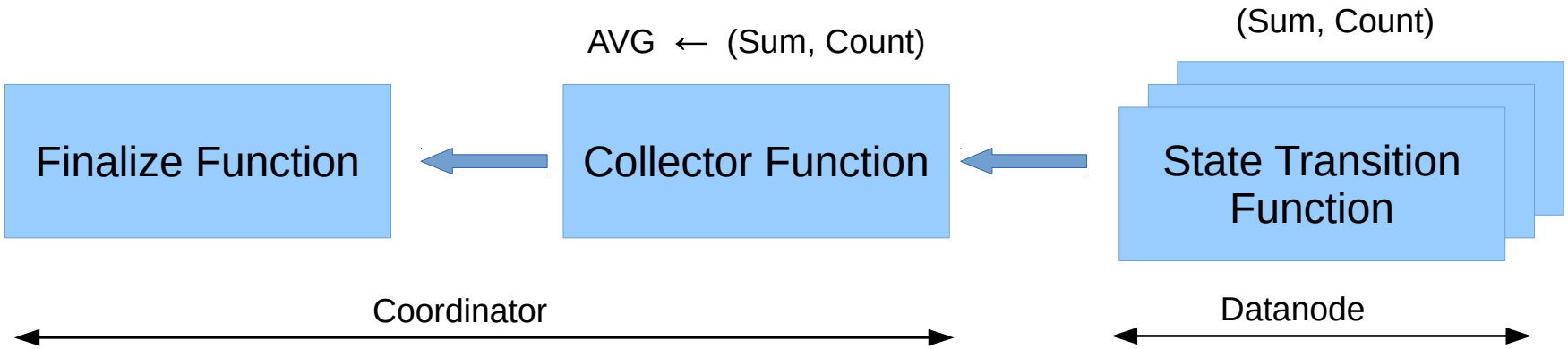
- Replicated Table and Partitioned Table
  - When the coordinator cannot determine which datanode to go from WHERE clause



Parallel Aggregation

Aggregate Functions in PostgreSQL

Aggregate Functions in Postgres-XC/XL



Similar to Map Reduce!

Specific statements

- CREATE BARRIER
  - Synchronize all node's WAL for restoration.
- CREATE|ALTER|DROP NODE
  - Maintenance of cluster node
    - Caution! – not automatically propagated. Issue to each coordinator.
- CREATE/DROP NODE GROUP
  - Alias for group of node
- EXECUTE DIRECT
  - Run SQL locally
  - Read operation only
    - If you are superuser, turn xc_maitenance_mode to on by set statement to allow write operations.
    - You must be responsible to any inconsistencies and side effects!

Specific catalogs

- pgxc_class
  - Definition of table distribution
- pgxc_node
  - Postgres-XC node information
- pgxc_group
  - Node group

Specific functions

- pgxc_version()
  - Show version
- pgxc_pool_check()
  - Check if connection pooler is consistent with pgxc_node catalog.
- pgxc_pool_reload
  - Reload cached connection data and synchronize pooler connection information with pgxc_node.
- pgxc_lock_for_backup
  - Only for adding new nodes.
  - Locks DDL execution to make catalog stable for backup and copy to new node.

Specific statements, catalogues, functions and parameters

http://postgres-x2.github.io/reference/1.2/html/sql-commands.html
for details

Specific parameters (planner parameters not included)

- gtm_backup_barrier (bool)
  - Enable CREATE BARRIER statement.
- persistent_datanode_connections (bool)
  - If “true”, session never releases connections.
- xc_maintenance_mode
  - Enable write operation in “EXECUTE DIRECT” statement.
  - Only allowed to root users.
- min_pool_size
  - Threashold for pooler to create new connection.
- max_pool_size
  - Max pooled connection size.
- pooler_port
  - Port number for the pooler (pgxc_ctl takes care of it)
- gtm_port
  - GTM port number (pgxc_ctl takes care of it)

Specific parameters (cont.)

- max_datanodes
- max_coordinators
- pgxcnode_cancel_delay
  - Timeout to wait cancel operation in millisconds.
  - Mainly for automatic test.
- gtm_host
  - GTM host name/IP address. Pgxc_ctl takes care of this.
- pgxc_node_name
  - Node name of the self. Pgxc_ctl takes care of this.

Community status and future

Specific statements

- CREATE/DROP NODE GROUP
  - Alias for group of node
- Unified again?

XC and XL community

- Postgres-XC is the original community
  - Based upon PostgreSQL 9.3
  - Tested more for OLPT workload
  - Now community activity as Postgres-X2
  - Stabilization
    - Participated by many Chinese engineers
    - Next minor release are planned in this August
- Postgres-XL was became separate community for more product-oriented and better stability
  - Based upon PostgreSQL 9.2
  - Shares most of XC code base
  - Tested more for OLAP workload
    - Direct data capture between datanodes
  - Provide many fixes. Most of them apply to XL as well
  - Just finished merge with Postgres 9.5 alfa
- Unified again?

Product status

- Source code inherits all the PostgreSQL repository (at some point)
- Fundamental features are all available
  - Global transaction management
  - SQL statements
  - Utilities
- Further challenges
  - Subtransaction (needed for full function support)
  - Catching up PostgreSQL (needed?)
  -

XC and XL community

- Both communities need much more resource to move forward
  - Developer
  - Tester
  - Real workload
- Now several Chinese farms are working together.
  - Much more active members are welcome!

XC and XL community

- Both communities need much more resource to move forward
  - Developer
  - Tester
  - Real workload
- Now several Chinese farms are working together.
  - Much more active members are welcome!

XC and XL community sites

Postgres-XC

https://github.com/postgres-x2
https://postgres-x2.github.io
https://groups.google.com/forum/#!forum/postgres-x2-dev
https://groups.google.com/forum/#!forum/postgres-x2-general

koichi.dbms@gmail.com
galylee@gmail.com

Postgres-XL

http://www.postgres-xl.org/

Configuring Postgres-XC

Pgxc_ctl

- Postgres-XC contrib module
- Postgers-XC configuration and operation tool
  - A kind of Postres-XC shell
  - Builtin commands
  - Can invoke any bash commands
    - Does not expand $(variable).
- Simple configuration
- Avoid many pitfalls in manual configuration and operation
- Bash-based configuration file
- You can write your favorite bash-script for your configuration

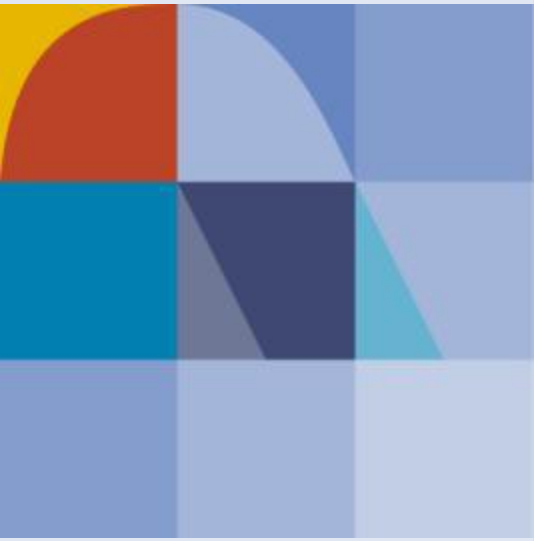
Pgxc_ctl builtin commands (major ones)

- prepare
  - Creates configuration file template
- deploy
  - Deploys postgres-xc binaries to necessary nodes
- Init [all]
  - Initialize postgres-xc cluster
    - Run initdb and initgtm at necessary nodes
    - Do additional configuration
    - Initialize node configuration
- Start/stop
  - Cluster and node start/stop
- Clean
  - Cleanup existing resource
- Monitor
  - See what node is running

Pgxc_ctl builtin commands (major ones)

- Createdb
  - Similar to createdb but select one coordinator to do it.
- Psql
  - Similar to psql but select one coordinator or specify coordinator name to connect to.
- Add
  - Add gtm_proxy, coordinator and datanode (master and slave)
- Remove
  - Remove gtm_proxy, coordinator and datanode (master and slave)

Demonstration



NTT DATA

Global IT Innovator