DTCC 2015

Database Technology
Conference China 2015

Postgres-XL/XC
Scale-out Approach in PostgreSQL

May 18th, 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

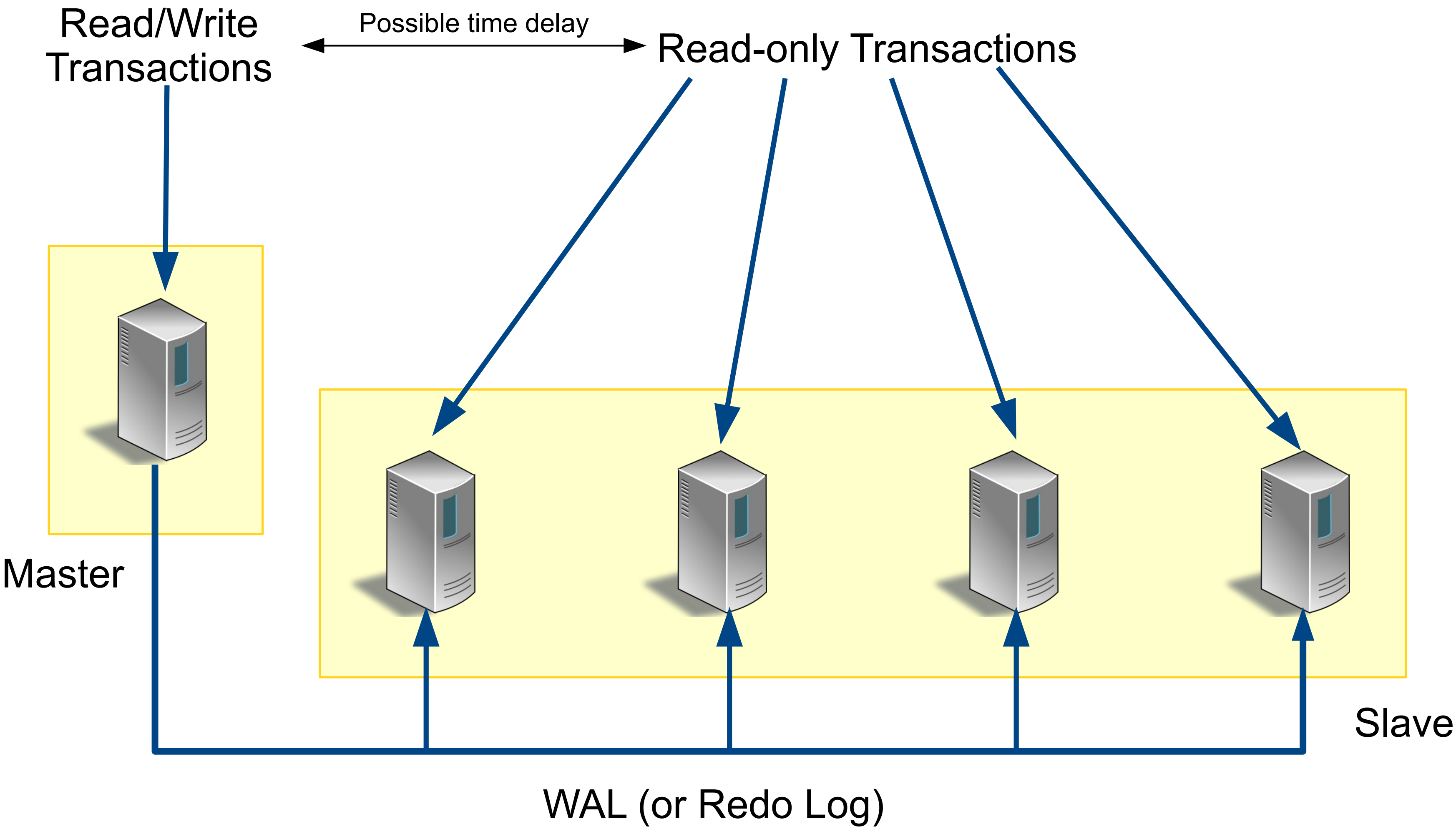
Motivation

- Growing Database Workload both for 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 on transaction ACID properties and SQL language.

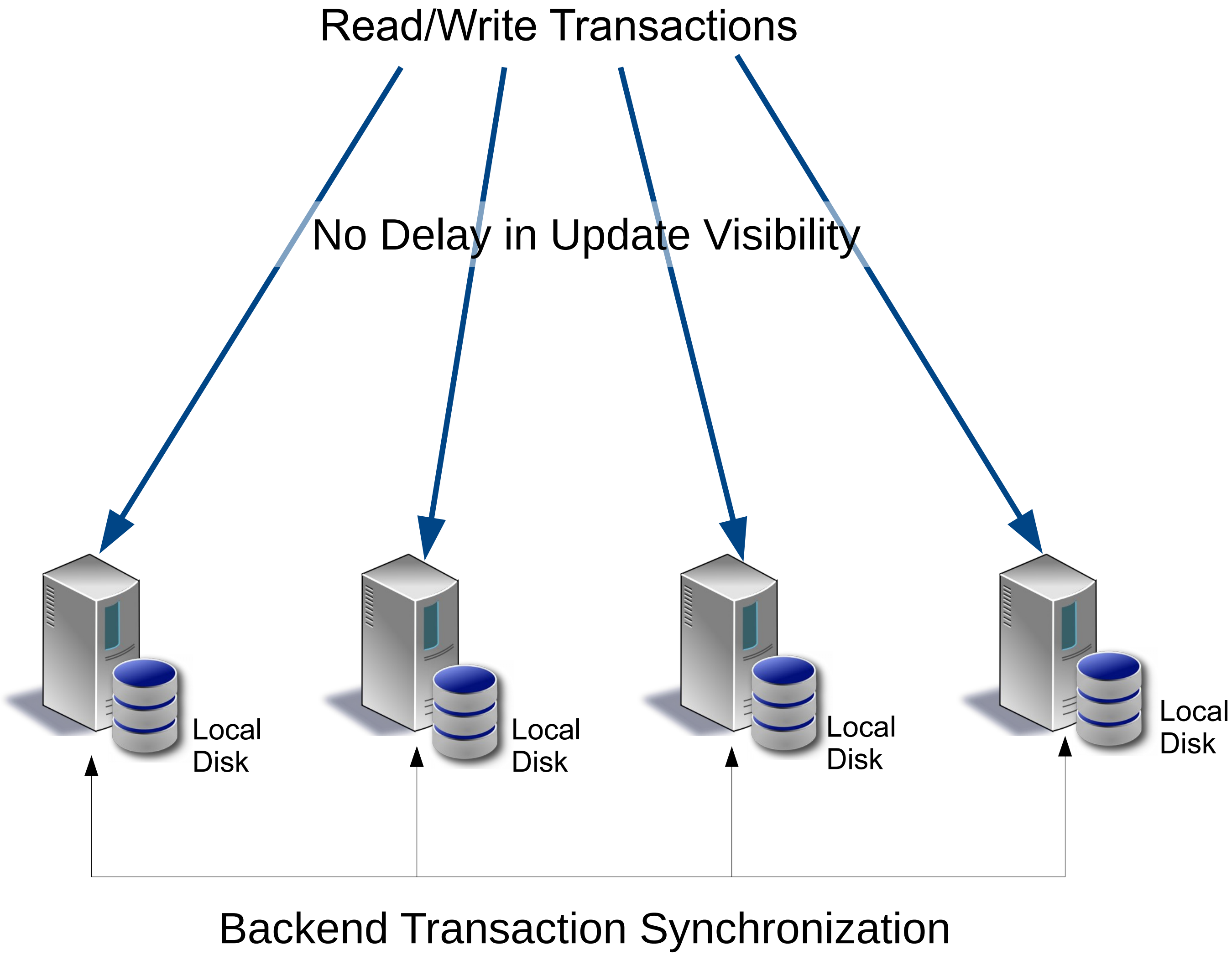
Scale-out approach

- 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

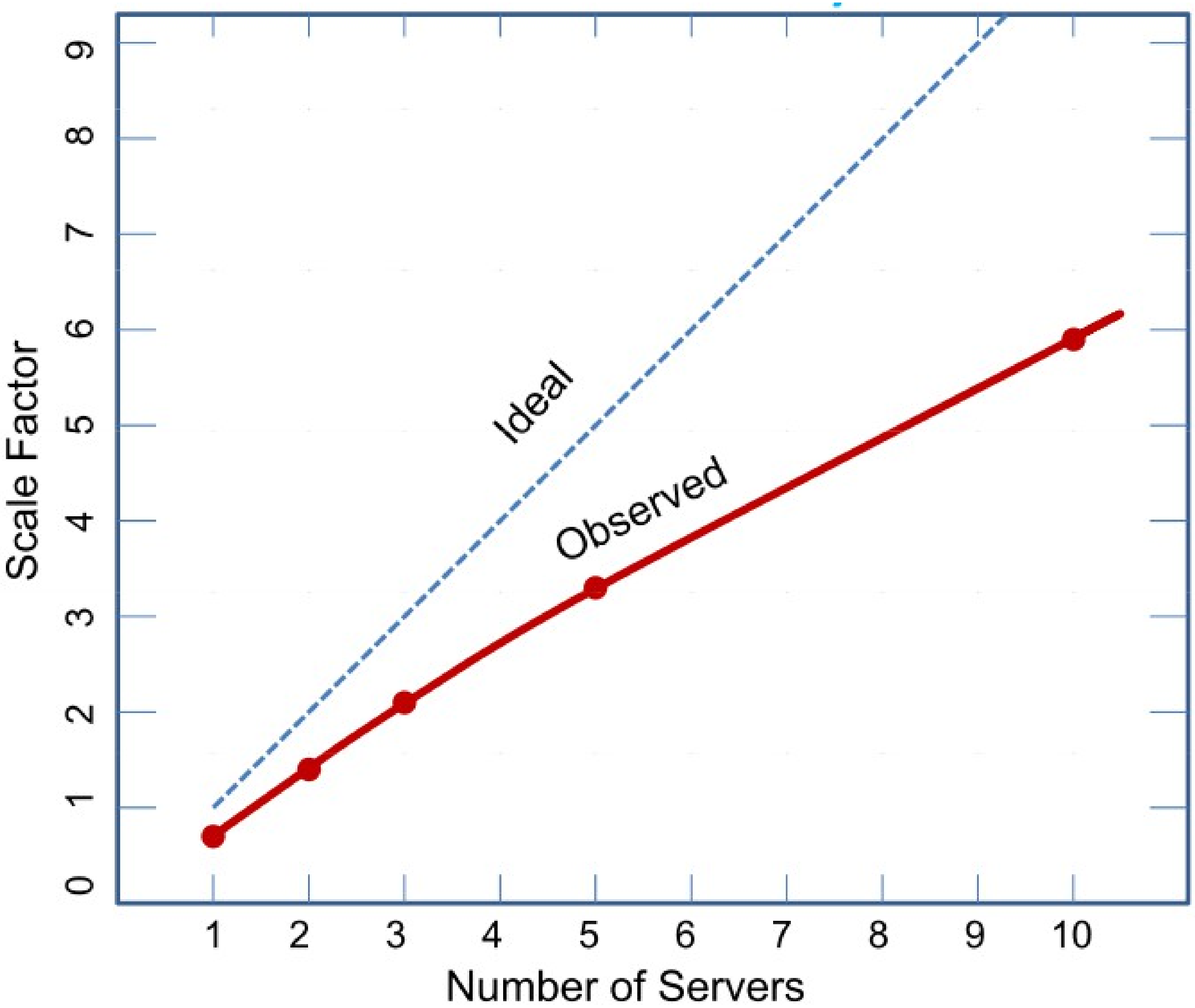


Scaling Out in Postgres XL/XC


Read/Write Transactions
No Delay in Update Visibility
Local
Disk
Local
Disk
Local
Disk
Local
Disk
Backend Transaction Synchronization

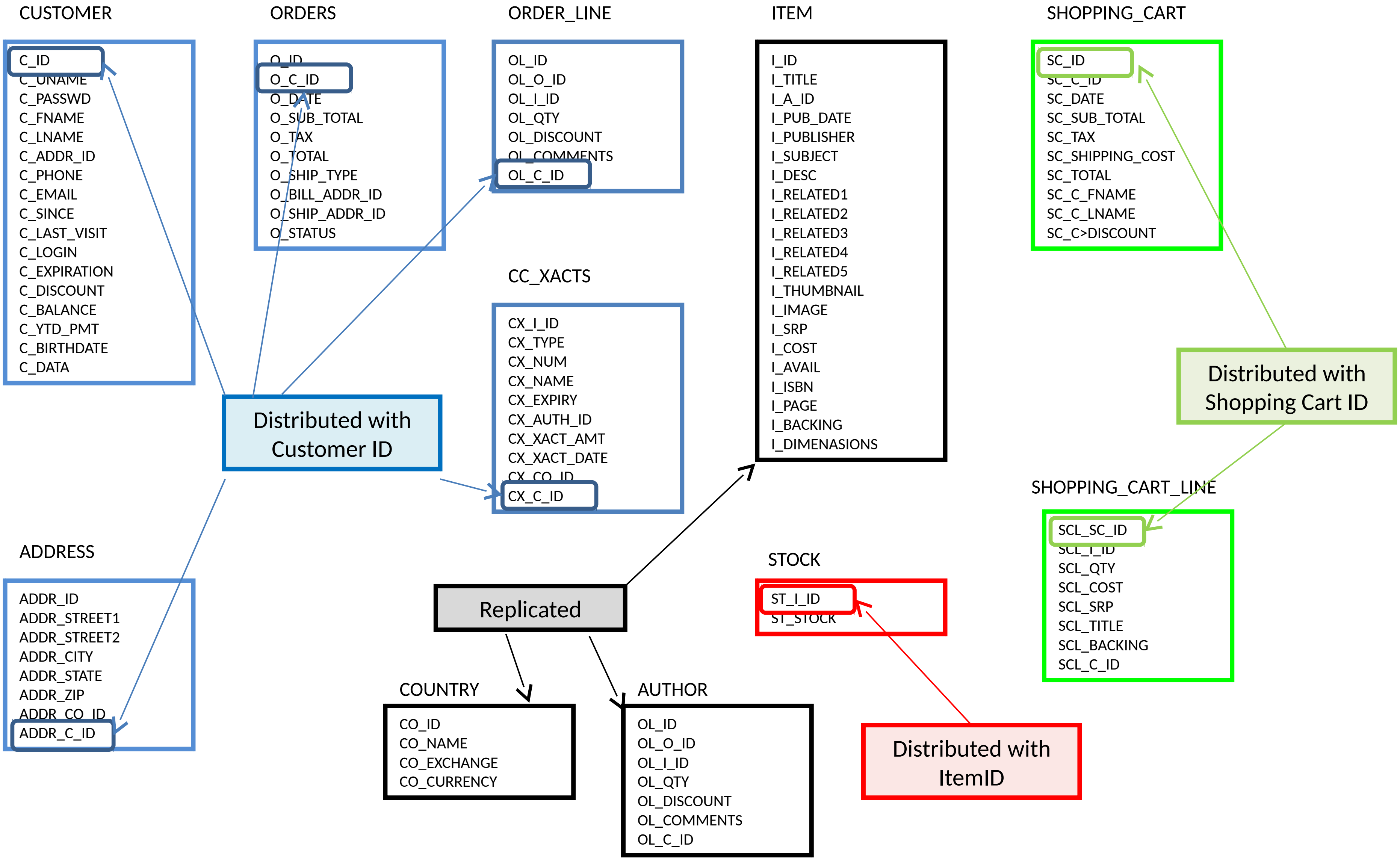
OLTP Workload Scalability and Table Design

DBT-1 Workload Scalability



DBT-1 (Rev)

Table Design in DBT-1 Benchmark



CUSTOMER

C_ID
C_UNAME
C_PASSWD
C_FNAME
C_LNAME
C_ADDR_ID
C_PHONE
C_EMAIL
C_SINCE
C_LAST_VISIT
C_LOGIN
C_EXPIRATION
C_DISCOUNT
C_BALANCE
C_YTD_PMT
C_BIRTHDATE
C_DATA

ORDERS

O_ID
O_C_ID
O_DATE
O_SUB_TOTAL
O_TAX
O_TOTAL
O_SHIP_TYPE
O_BILL_ADDR_ID
O_SHIP_ADDR_ID
O_STATUS

ORDER_LINE

OL_ID
OL_O_ID
OL_I_ID
OL_QTY
OL_DISCOUNT
OL_COMMENTS
OL_C_ID

ITEM

I_ID
I_TITLE
I_A_ID
I_PUB_DATE
I_PUBLISHER
I_SUBJECT
I_DESC
I_RELATED1
I_RELATED2
I_RELATED3
I_RELATED4
I_RELATED5
I_THUMBNAIL
I_IMAGE
I_SRP
I_COST
I_AVAIL
I_ISBN
I_PAGE
I_BACKING
I_DIMENASIONS

SHOPPING_CART

SC_ID
SC_C_ID
SC_DATE
SC_SUB_TOTAL
SC_TAX
SC_SHIPPING_COST
SC_TOTAL
SC_C_FNAME
SC_C_LNAME
SC_C>DISCOUNT

CC_XACTS

CX_I_ID
CX_TYPE
CX_NUM
CX_NAME
CX_EXPIRY
CX_AUTH_ID
CX_XACT_AMT
CX_XACT_DATE
CX_CO_ID
CX_C_ID

Distributed with Customer ID

Distributed with Shopping Cart ID

SHOPPING_CART_LINE

SCL_SC_ID
SCL_I_ID
SCL_QTY
SCL_COST
SCL_SRP
SCL_TITLE
SCL_BACKING
SCL_C_ID

ADDRESS

ADDR_ID
ADDR_STREET1
ADDR_STREET2
ADDR_CITY
ADDR_STATE
ADDR_ZIP
ADDR_CO_ID
ADDR_C_ID

Replicated

STOCK

ST_I_ID
ST_STOCK

COUNTRY

CO_ID
CO_NAME
CO_EXCHANGE
CO_CURRENCY

AUTHOR

OL_ID
OL_O_ID
OL_I_ID
OL_QTY
OL_DISCOUNT
OL_COMMENTS
OL_C_ID

Distributed with ItemID

Scale Out Approach (1): Table Distribution/Replication

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

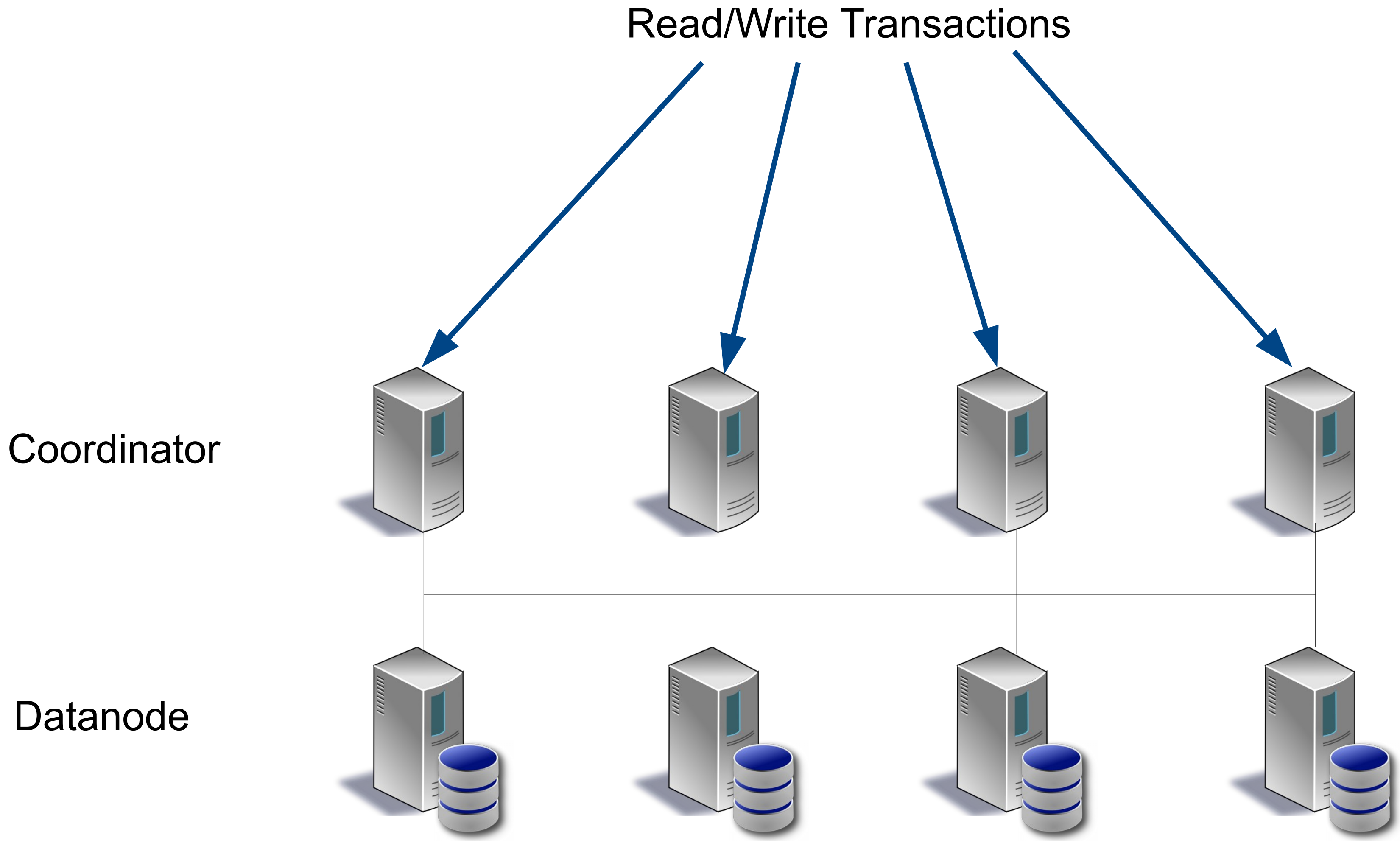
Coordinator:

- Holds catalog information
- Build global SQL plan
- Tell Datanode what to do in SQL statements

Datanode

- Holds actual data
- Run local SQL statement from Coordiator
  (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?

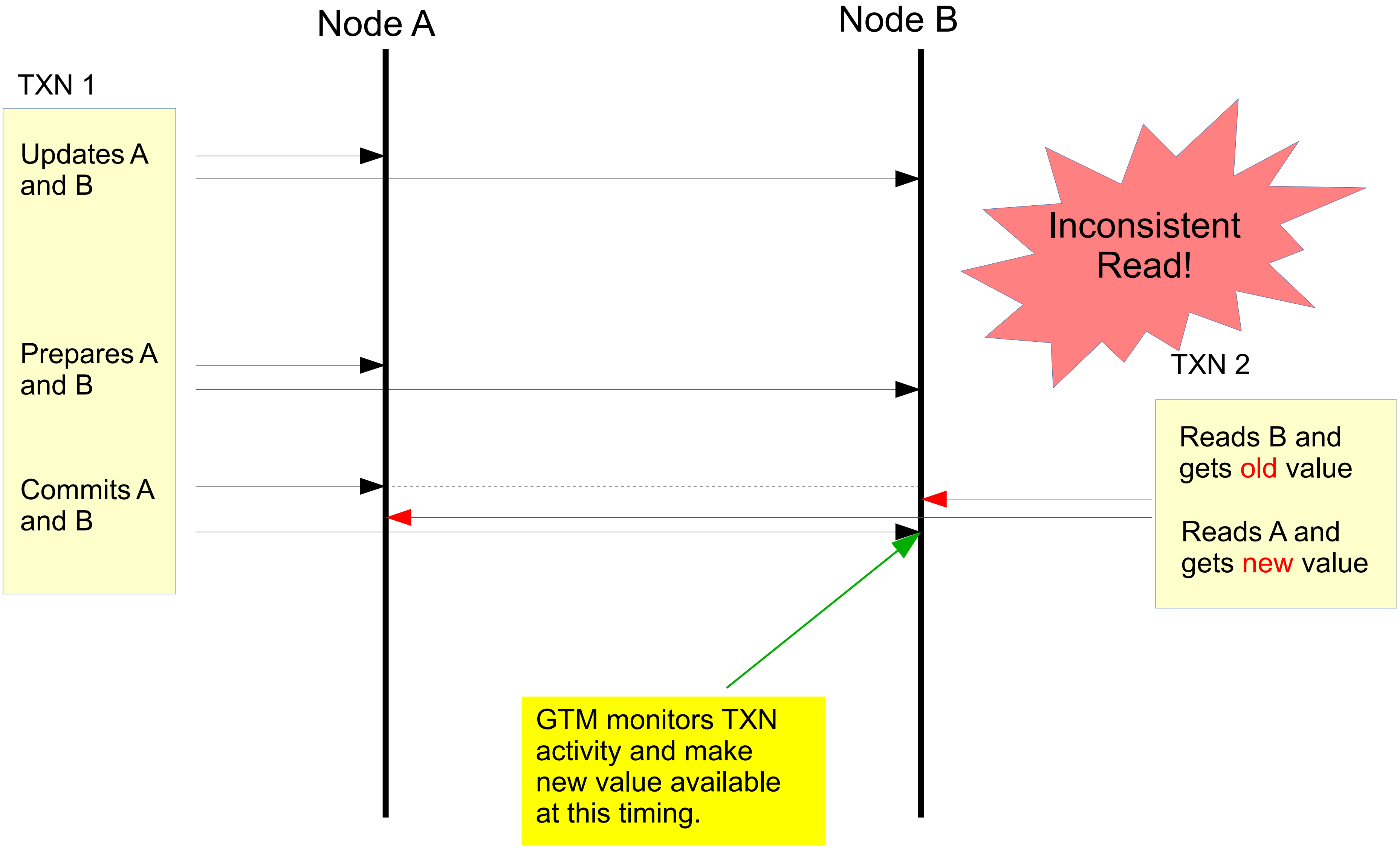
Two-Phase Commit Protocol Does:

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

Two-Phase Commit Protocol Doesn't:

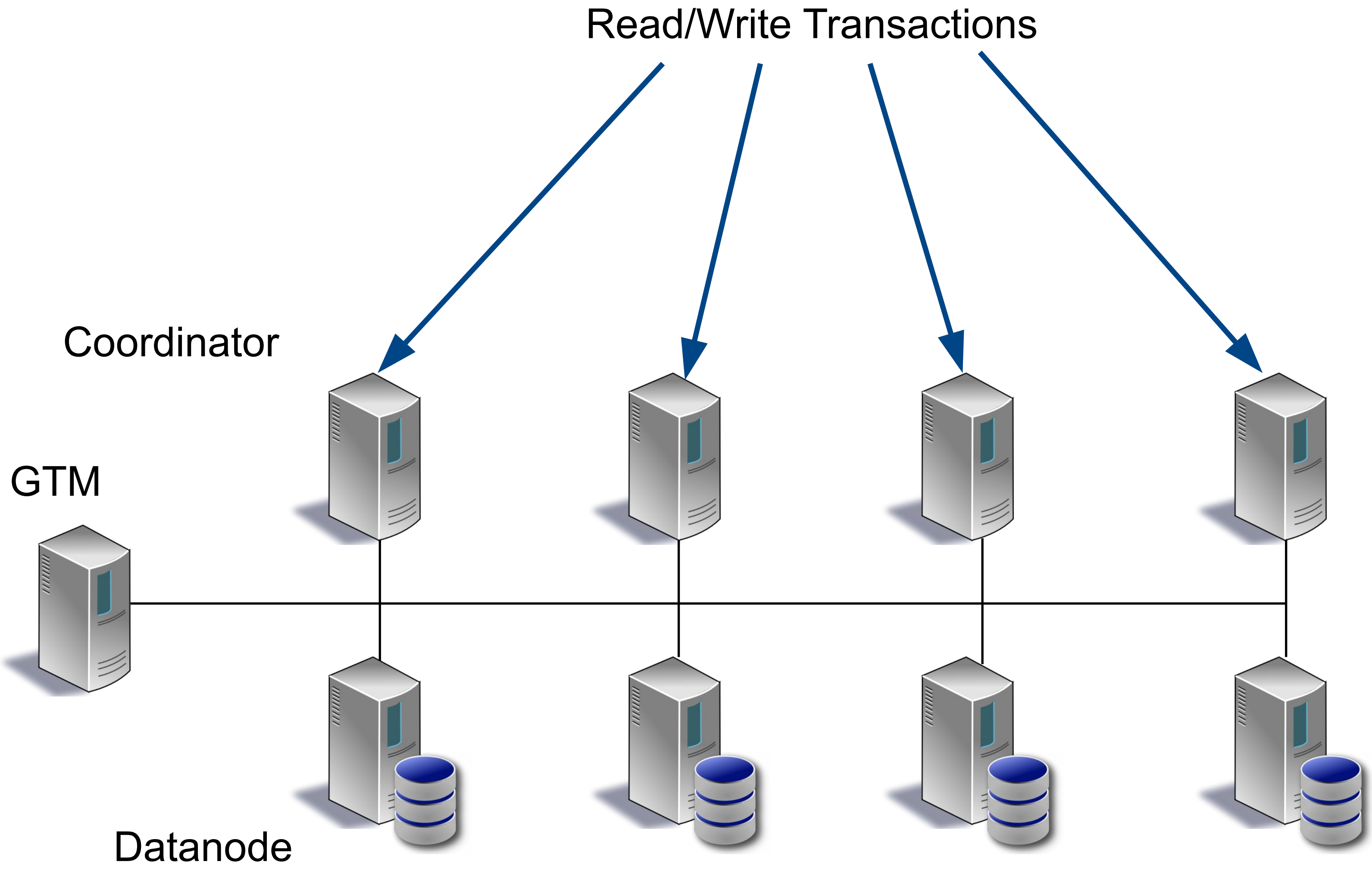
- Maintain Atomic Visibility of Updates to other transactions (next slide)

Atomic Visibility and GTM



GTM monitors TXN activity and make new value available at this timing.

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

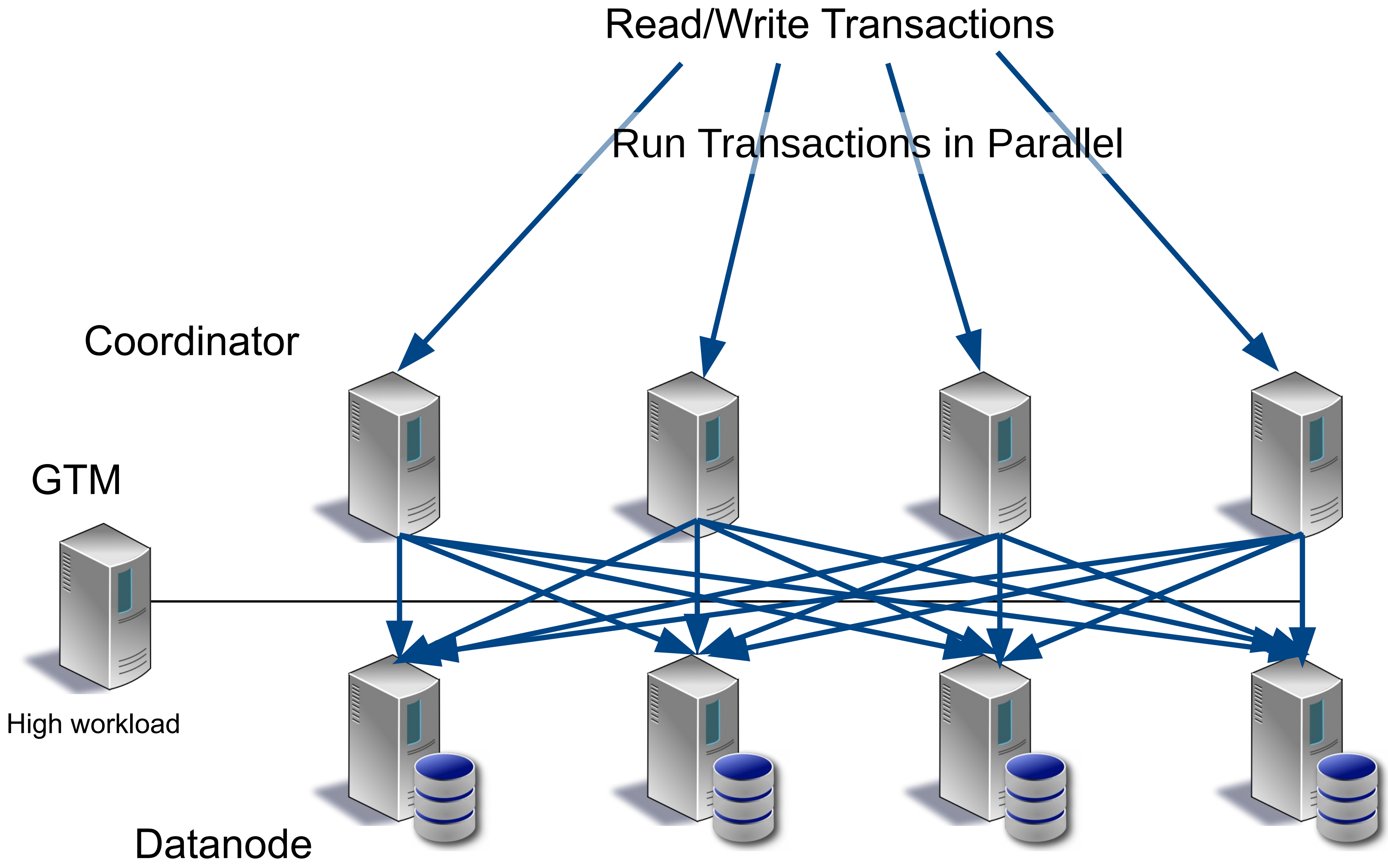
Number of Transactions: Many

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

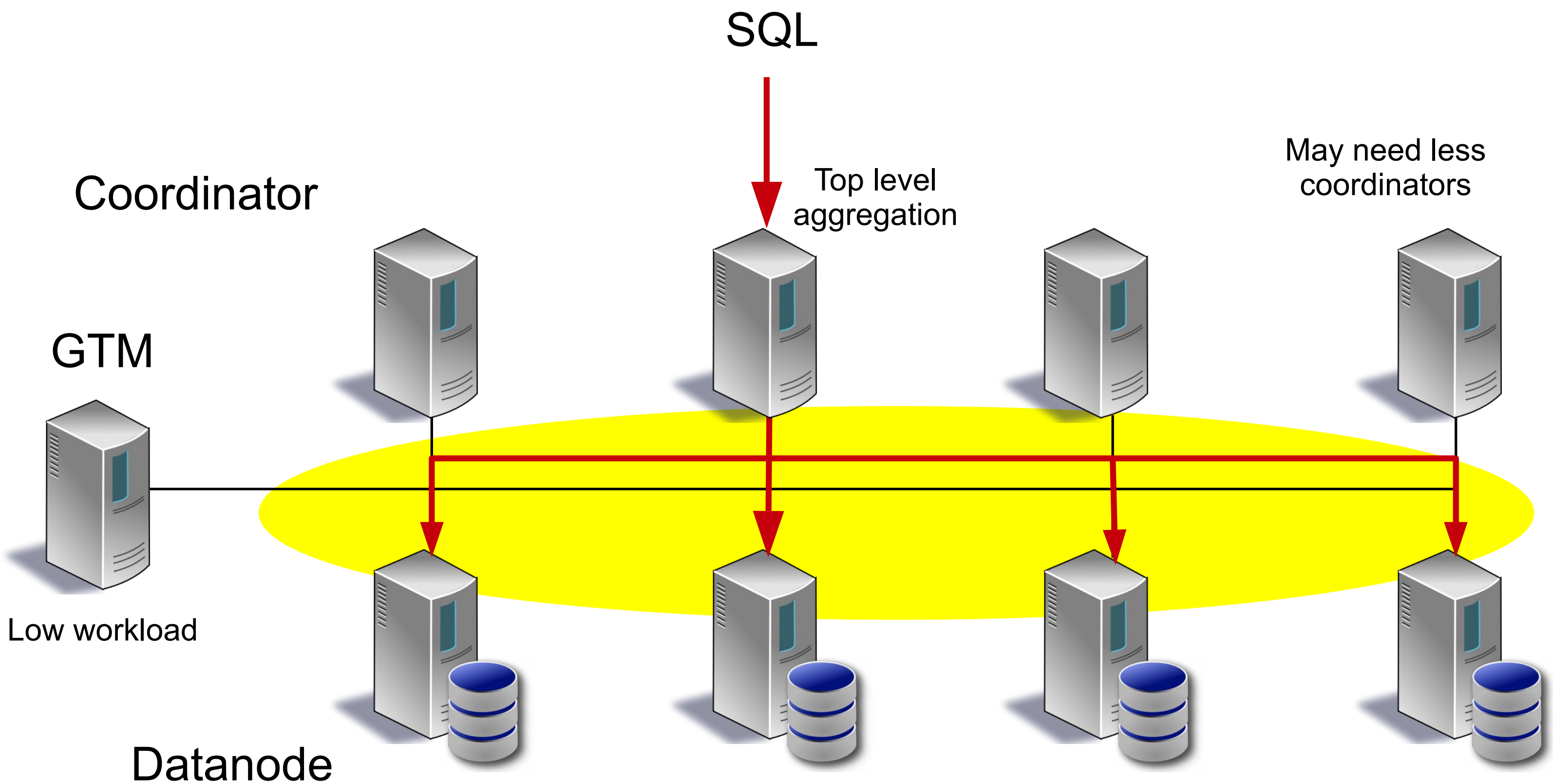
Number of Transactions: Small

Number of Involved Table Rows: Huge

Locality of Row Allocation: Low

Update Frequency: Low

Scaling Out OLAP Workload

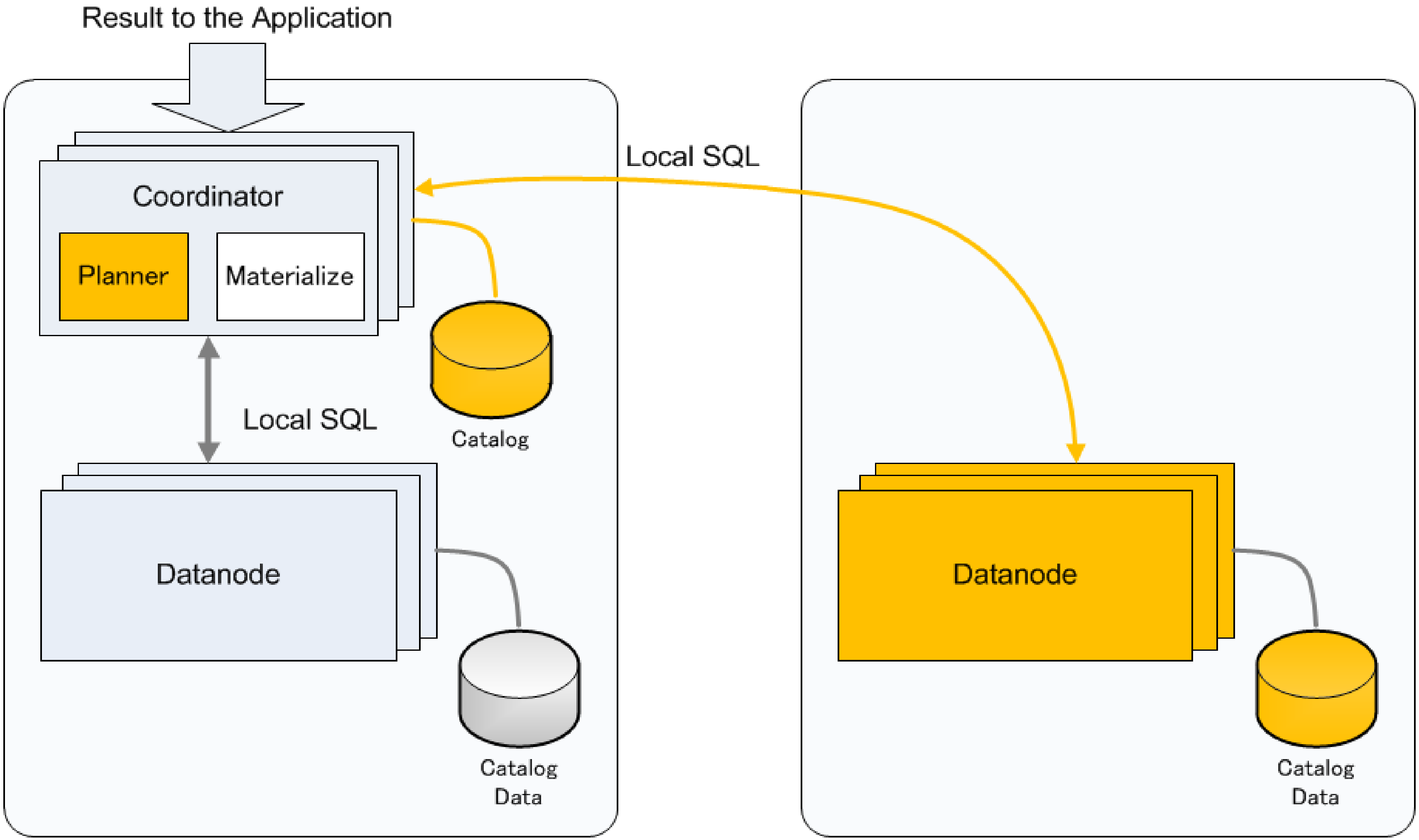


Run Small Local SQLs for each Datanode in Parallel

Join Offloading

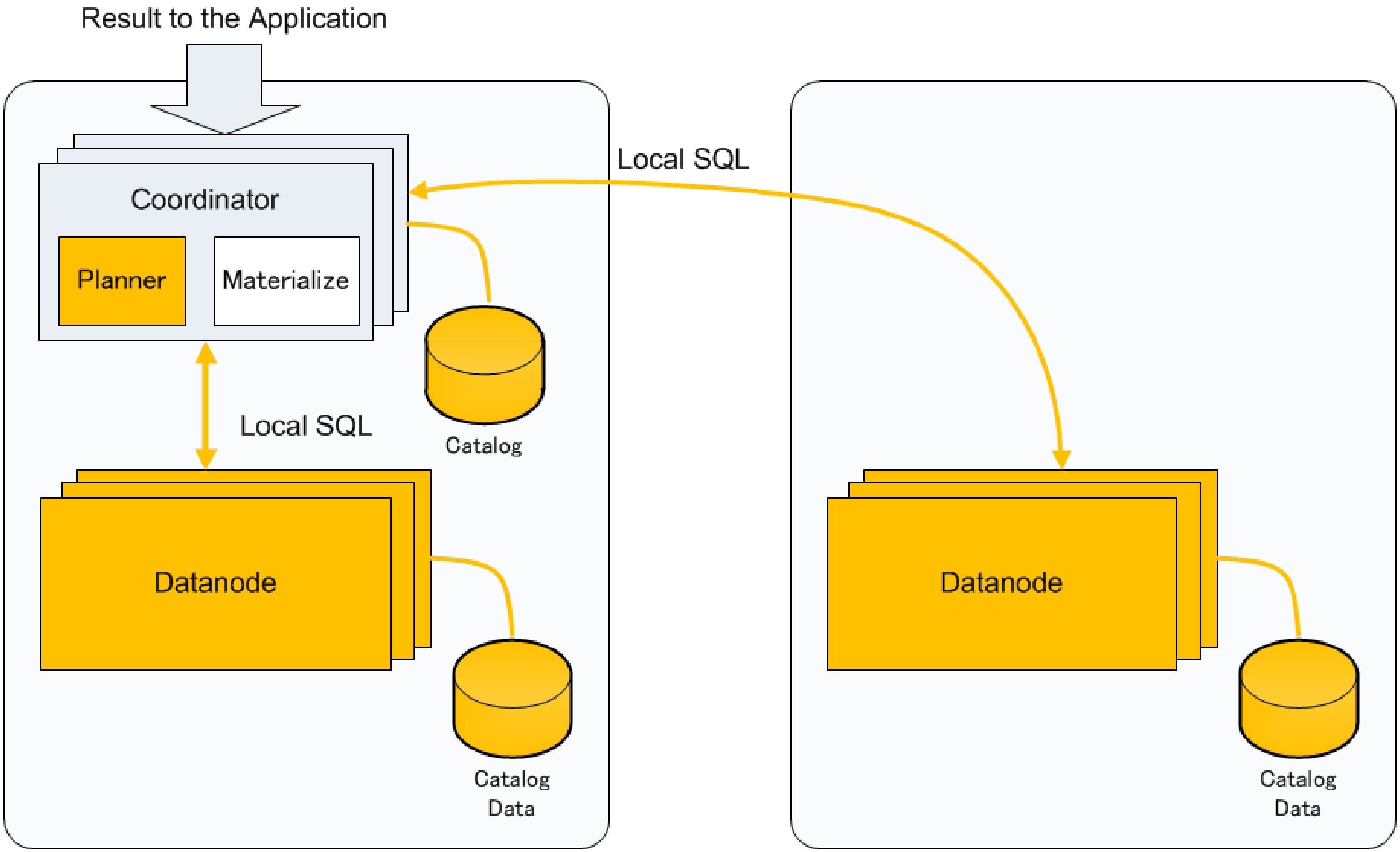
Join Offloading: When row allocation is available

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



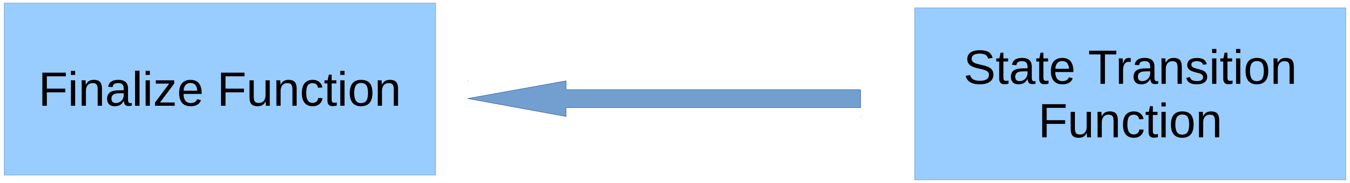
Join Offloading: When row allocation is available

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

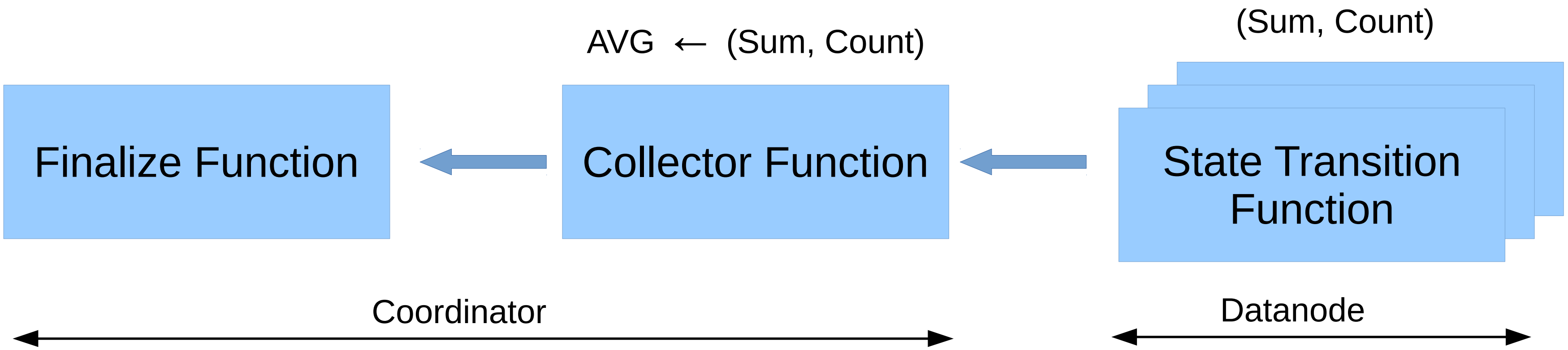


Parallel Aggregation

Aggregate Functions in PostgreSQL



Aggregate Functions in Postgres-XC/XL



Similar to Map Reduce!

Community status and future

XC and XL community

- Postgres-XC is the original community
  - Based upon PostgreSQL 9.3
  - Tested more for OLPT workload
- 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
- 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

XC and XL community sites

Postgres-XC

http://sourceforge.net/projects/postgres-xc/

koichi.dbms@gmail.com

Postgres-XL

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

NTT DATA

Global IT Innovator