Postgres-XC Dynamic Cluster Management

Koichi Suzuki
Postgres-XC Development Group

Postgres Open 2013
September 17th, 2013
Hotel Sax, Chicago, USA

Outline of the Talk

- Postgres-XC short review
  - Architecture and scalability
- Configurating Postgres-XC
  - Manual configuration
  - Dedicated tools
- Operating Postgres-XC
- Postgres-XC node management
  - Dynamic node addition/removal
- Failure handling and HA
- Current status and future schedule

Self Introduction

- Postgres-XC leader and core architect, as well as a core developer
  - Whole architecture design
  - Global transaction management and data distribution as a key for write-scalability
- Work for NTT DATA Intellilink
  - Subsidiary of NTT DATA corporation dedicated for system platform
  - Member of NTT group company
- Resources
  - koichi.clarinet@gmail.com (facebook, linkedin)
  - @koichiclarinet (twitter)

Sept. 18th, 2013

Postgres-XC Short Review

Useful Materials

- Postgres-XC
  - http://postgres-xc.sourceforge.net/ (project web site)
  - http://sourceforge.net/projects/postgres-xc/ (development site)
  - http://postgres-xc.sourceforge.net/misc-docs/20120614_PGXC_Tutorial_global.pdf (Postgres-XC tutorial)
  - http://postgres-xc.sourceforge.net/misc-docs/Prague_Presentation_20121024.pdf (HA architecture and feature)
  - http://postgresxc.wikia.com/wiki/File:Postgres-XC_20110711_01.pdf (General architecture and feature)
- PostgreSQL resource agents for Pacemaker/Heartbeat
  - sf-ex : mount filesystem exclusively
    - https://github.com/ClusterLabs/resource-agents/blob/master/heartbeat/sfex
  - postgres – streaming replication
    - https://github.com/ClusterLabs/resource-agents/blob/master/heartbeat/pgsql

Postgres-XC in Postgres Open

- 2011
  - Postgres-XC inroduction
  - Architecture, scalability
- 2012
  - Postgres-XC internals
  - SQL planning and execution
- 2013
  - Deployment, configuration, operation
  - Slaves for high-availability
  - Cluster management

Postgres-XC 1.1

- Dynamic node addition/removal
  - Related new options to initidb, pg_dump, and pg_dumpall
  - DDL lock during node addition/removal
  - Table re-distribution
- Row-level triggers
- RETURNING
- pgxc_ctl operation tool
- Many distributed SQL planner improvements
  - Outer joins
  - LIMIT, ORDER BY, GROUP BY pushdown
- Executor improvement
  - Merge and hash-join at the coordinator
  - Distributed sort and merge at datanodes
- Others
  - PostgreSQL 9.2.4 merge
  - GTM restore point backup

Configuring Postgres-XC

Manual Operation

- Configure postgresql.conf for all the components
  - Coordinator
  - Datanode
- Configure node configuration by CREATE|ALTER NODE
  - Has to repeat for all the coordinators

- Could be very painful
- Operation such as Start/Stop/Failover/Add nodes/Remove nodes could be complicated too

Overview of pgxc_ctl

- Command line tool
- Takes care of Postgres-XC database cluster operation
  - Configuration/Initialization
  - Start/stop
  - Simple monitoring
  - Failover
  - Add/remove components dynamically.

Configuration (1)

- Configuration file is a bash-script
  - Easy to customize
  - Flexible configuration definitions

```
# pgxcInstallDir variable is needed if you invoke "deploy" command from pgxc_ctl utility.
# If don't you don't need this variable.
pgxcInstallDir=$HOME/pgxc
#---- OVERALL -----------------------------------------------------------------------------
#
pgxcOwner=koichi              # owner of the Postgres-XC databaseo cluster.  Here, we use this
                              # both as linus user and database user.  This must be
                              # the super user of each coordinator and datanode.
pgxcUser=$pgxcOwner           # OS user of Postgres-XC owner

tmpDir=/tmp                   # temporary dir used in XC servers
localTmpDir=$tmpDir           # temporary dir used here locally

configBackup=n                # If you want config file backup
configBackupHost=yourBackupHost # host to backup config file
configBackupDir=yourBackupDir   # Backup directory
configBackupFile=yourBackupFile # Backup file name --> Need to synchronize when original changed.

#---- GTM ------------------------------------------------------------------------------------

#---- Overall -------
gtmName=gtm

#---- GTM Master -----------------------------------------------

#---- Overall ----
gtmMasterServer=node13
gtmMasterPort=20001
gtmMasterDir=$HOME/pgxc/nodes/gtm
```

Sept. 18th, 2013

Configuration(2)

- Each component needs
  - Nodename
  - Hostname/address
  - Port
  - Work Directory
  - (Coordinator needs additional pooler port)
  - (Coordinator/Datanode slave needs WAL archive directory)
  - Specific configuration parameters.
- You can specify these configuration for each component
- Pgxc_ctl will take care of everything else.
- Example will be given later.

Demonstration

- Configuration file template
  - prepare command
- Configuration/Initialization
  - clean command
  - init command
- Start/stop
  - start command
  - stop command
  - kill command
- Selecting coordnator automatically
  - Createdb
  - Createuser
  - Dropdb
  - Psql
- Failover
  - failover command
  - reconnect command
- Add/remove components dynamically
  - Add command
  - Remove command
- Table re-distribution
  - ALTER TABLE statement

Configuration Specifics

- You should specify
  - GTM master (mandatory)
  - GTM slave (optional)
  - GTM Proxy (optional, needed for HA)
  - Coordinator master (at least one)
  - Coordinator slave (optional)
  - Datanode master (at least one)
  - Datanode slave (optional)

Overall Configuration

```
# pgxcInstallDir variable is needed if you invoke "deploy" command from pgxc_ctl utility.
# If don't you don't need this variable.
pgxcInstallDir=$HOME/pgxc
#---- OVERALL -----------------------------------------------------------------------------
#
pgxcOwner=koichi             # owner of the Postgres-XC databaseo cluster.  Here, we use this
                             # both as linus user and database user.  This must be
                             # the super user of each coordinator and datanode.
pgxcUser=$pgxcOwner          # OS user of Postgres-XC owner

tmpDir=/tmp                  # temporary dir used in XC servers
localTmpDir=$tmpDir          # temporary dir used here locally

configBackup=n               # If you want config file backup
configBackupHost=yourBackupHost  # host to backup config file
configBackupDir=yourBackupDir    # Backup directory
configBackupFile=yourBackupFile  # Backup file name --> Need to synchronize when original changed.
```

GTM Master

```
#---- GTM ----------------------------------------------------------------------------------

# GTM is mandatory.  You must have at least (and only) one GTM master in your Postgres-XC cluster.
# If GTM crashes and you need to reconfigure it, you can do it by pgxc_update_gtm command to update
# GTM master with others.   Of course, we provide pgxc_remove_gtm command to remove it.  This command
# will not stop the current GTM.  It is up to the operator.

#---- Overall -------
gtmName=gtm

#---- GTM Master -----------------------------------------------

#---- Overall ----
gtmMasterServer=node13
gtmMasterPort=20001
gtmMasterDir=$HOME/pgxc/nodes/gtm

#---- Configuration ---
gtmExtraConfig=none        # Will be added gtm.conf for both Master and Slave (done at initilization only)
gtmMasterSpecificExtraConfig=none   # Will be added to Master's gtm.conf (done at initialization only)
```

GTM Slave

```
#---- GTM Slave -----------------------------------------------

# Because GTM is a key component to maintain database consistency, you may want to configure GTM slave
# for backup.

#---- Overall ------
gtmSlave=y                  # Specify y if you configure GTM Slave.   Otherwise, GTM slave will not be configured and
                            # all the following variables will be reset.
gtmSlaveServer=node12       # value none means GTM slave is not available.
                            # Give none if you don't configure GTM Slave.
GtmSlavePort=20001          # Not used if you don't configure GTM slave.
gtmSlaveDir=$HOME/pgxc/nodes/gtm    # Not used if you don't configure GTM slave.

# Please note that when you have GTM failover, then there will be no slave available until you configure the slave
# again. (pgxc_add_gtm_slave function will handle it)

#---- Configuration ----
gtmSlaveSpecificExtraConfig=none    # Will be added to Slave's gtm.conf (done at initialization only)
```

GTM Proxy

```
#---- GTM Proxy ------------------------------------------------------------------------------------------------
# GTM proxy will be selected based upon which server each component runs on.
# When fails over to the slave, the slave inherits its master's gtm proxy.  It should be
# reconfigured based upon the new location.
#
# To do so, slave should be restarted.   So pg_ctl promote -> (edit postgresql.conf and recovery.conf) -> pg_ctl restart
#
# You don't have to configure GTM Proxy if you dont' configure GTM slave or you are happy if every component connects
# to GTM Master directly.  If you configure GTL slave, you must configure GTM proxy too.

#---- Shortcuts ------
gtmProxyDir=$HOME/pgxc/nodes/gtm_pxy

#---- Overall -------
gtmProxy=y              # Specify y if you conifugre at least one GTM proxy.   You may not configure gtm proxies
                    # only when you dont' configure GTM slaves.
                    # If you specify this value not to y, the following parameters will be set to default empty values.
                    # If we find there're no valid Proxy server names (means, every servers are specified
                    # as none), then gtmProxy value will be set to "n" and all the entries will be set to
                    # empty values.
gtmProxyNames=(gtm_pxy1 gtm_pxy2 gtm_pxy3 gtm_pxy4)   # No used if it is not configured
gtmProxyServers=(node06 node07 node08 node09)          # Specify none if you dont' configure it.
gtmProxyPorts=(20001 20001 20001 20001)                # Not used if it is not configured.
gtmProxyDirs=($gtmProxyDir $gtmProxyDir $gtmProxyDir $gtmProxyDir)    # Not used if it is not configured.

#---- Configuration ----
gtmPxyExtraConfig=none      # Extra configuration parameter for gtm_proxy
gtmPxySpecificExtraConfig=(none none none none)
```

Sept. 18th, 2013

Coordinator Common

```
#---- Coordinators ---------------------------------------------------------------------------------------------

#---- shortcuts ----------
coordMasterDir=$HOME/pgxc/nodes/coord
coordSlaveDir=$HOME/pgxc/nodes/coord_slave
coordArchLogDir=$HOME/pgxc/nodes/coord_archlog

#---- Overall ------------
coordNames=(coord1 coord2 coord3 coord4)      # Master and slave use the same name
coordPorts=(20004 20005 20004 20005)          # Master and slave use the same port
poolerPorts=(20010 20011 20010 20011)         # Master and slave use the same pooler port
coordPgHbaEntries=(192.168.1.0/24)            # Assumes that all the coordinator (master/slave) accepts
                                              # the same connection
                                              # This entry allows only $pgxcOwner to connect.
                                              # If you'd like to setup another connection, you should
                                              # supply these entries through files specified below.
# Note: The above parameter is extracted as "host all all 0.0.0.0/0 trust".   If you don't want
# such setups, specify the value () to this variable and suplly what you want using coordExtraPgHba
# and/or coordSpecificExtraPgHba variables.
```

Coordinator Master

```
#---- Master -------------
coordMasterServers=(node06 node07 node08 node09)       # none means this master is not available
coordMasterDirs=($coordMasterDir $coordMasterDir $coordMasterDir $coordMasterDir)
CoordMaxWALsernder=5          # max_wal_senders: needed to configure slave. If zero value is specified,
                              # it is expected to supply this parameter explicitly by external files
                              # specified in the following.   If you don't configure slaves, leave this value to zero.
coordMaxWALSenders=($coordMaxWALsernder $coordMaxWALsernder $coordMaxWALsernder
$coordMaxWALsernder)
                              # max_wal_senders configuration for each coordinator.
```

Coordinator Slave

```
#---- Slave -------------
coordSlave=y            # Specify y if you configure at least one coordiantor slave.  Otherwise, the following
                        # configuration parameters will be set to empty values.
                        # If no effective server names are found (that is, every servers are specified as none),
                        # then coordSlave value will be set to n and all the following values will be set to
                        # empty values.
coordSlaveSync=y        # Specify to connect with synchronized mode.   At present, only “y” is assumed.
coordSlaveServers=(node07 node08 node09 node06)       # none means this slave is not available
coordSlaveDirs=($coordSlaveDir $coordSlaveDir $coordSlaveDir $coordSlaveDir)
coordArchLogDirs=($coordArchLogDir $coordArchLogDir $coordArchLogDir $coordArchLogDir)
```

Coordinator Extra Configuration

```
#---- Configuration files---
# Need these when you'd like setup specific non-default configuration
# These files will go to corresponding files for the master.
# You may supply your bash script to setup extra config lines and extra pg_hba.conf entries
# Or you may supply these files manually.
coordExtraConfig=coordExtraConfig   # Extra configuration file for coordinators.
                    # This file will be added to all the coordinators'
                    # postgresql.conf
# Pleae note that the following sets up minimum parameters which you may want to change.
# You can put your postgresql.conf lines here.
cat > $coordExtraConfig <<EOF
#================================================
# Added to all the coordinator postgresql.conf
# Original: $coordExtraConfig
log_destination = 'stderr'
logging_collector = on
log_directory = 'pg_log'
listen_addresses = '*'
max_connections = 100
EOF
```

Datanode Common

```
#---- Datanodes -------------------------------------------------------------------------------------------

#---- Shortcuts --------------
datanodeMasterDir=$HOME/pgxc/nodes/dn_master
datanodeSlaveDir=$HOME/pgxc/nodes/dn_slave
datanodeArchLogDir=$HOME/pgxc/nodes/datanode_archlog

#---- Overall ---------------
# At present, xc has a priblem to issue ALTER NODE against the primay node.  Until it is fixed, the test will be done
# without this feature.
primaryDatanode=datanode1                       # Primary Node.
datanodeNames=(datanode1 datanode2 datanode3 datanode4)
datanodePorts=(20008 20009 20008 20009)  # Master and slave use the same port!
datanodePgHbaEntries=(192.168.1.0/24)         # Assumes that all the coordinator (master/slave) accepts
                                                # the same connection
                                                # This list sets up pg_hba.conf for $pgxcOwner user.
                                                # If you'd like to setup other entries, supply them
                                                # through extra configuration files specified below.
# Note: The above parameter is extracted as "host all all 0.0.0.0/0 trust".   If you don't want
# such setups, specify the value () to this variable and suplly what you want using datanodeExtraPgHba
# and/or datanodeSpecificExtraPgHba variables.
```

Sept. 18th, 2013

Datanode Master

```
#---- Master ----------------
datanodeMasterServers=(node06 node07 node08 node09)
                                        # none means this master is not available.
                                        # This means that there should be the master but is down.
                                        # The cluster is not operational until the master is
                                        # recovered and ready to run.
datanodeMasterDirs=($datanodeMasterDir $datanodeMasterDir $datanodeMasterDir $datanodeMasterDir)
datanodeMaxWalSender=5                  # max_wal_senders: needed to configure slave. If zero value is
                                        # specified, it is expected this parameter is explicitly supplied
                                        # by external configuration files.
                                        # If you don't configure slaves, leave this value zero.
datanodeMaxWALSenders=($datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender
$datanodeMaxWalSender)
                                        # max_wal_senders configuration for each datanode
```

Datanode Slave

```
#---- Slave -----------------
datanodeSlave=y         # Specify y if you configure at least one coordiantor slave.  Otherwise, the following
                        # configuration parameters will be set to empty values.
                        # If no effective server names are found (that is, every servers are specified as none),
                        # then datanodeSlave value will be set to n and all the following values will be set to
                        # empty values.
datanodeSlaveServers=(node07 node08 node09 node06)  # value none means this slave is not available
datanodeSlaveSync=y     # If datanode slave is connected in synchronized mode
datanodeSlaveDirs=($datanodeSlaveDir $datanodeSlaveDir $datanodeSlaveDir $datanodeSlaveDir)
datanodeArchLogDirs=( $datanodeArchLogDir $datanodeArchLogDir $datanodeArchLogDir
$datanodeArchLogDir )
```

Datanode Additional Configuration

```
# ---- Configuration files ---
# You may supply your bash script to setup extra config lines and extra pg_hba.conf entries here.
# These files will go to corresponding files for the master.
# Or you may supply these files manually.
datanodeExtraConfig=none        # Extra configuration file for datanodes.  This file will be added to all the
                                # datanodes' postgresql.conf
datanodeSpecificExtraConfig=(none none none none)
datanodeExtraPgHba=none         # Extra entry for pg_hba.conf.  This file will be added to all the
                                #datanodes' postgresql.conf
datanodeSpecificExtraPgHba=(none none none none)
```

Sept. 18th, 2013

When configuration changes

- Failover by components failure
- Add/remove componentsc

- Configuration file is modified.
- Backed up if configured so.

```
#==================================================
# pgxc configuration file updated due to GTM failover
#       20130704_16:56:42
gtmMasterServer=node12
gtmMasterPort=20001
gtmMasterDir=/home/koichi/pgxc/nodes/gtm
gtmSlave=n
gtmSlaveServer=none
gtmSlavePort=0
gtmSlaveDir=none
#----End of reconfiguration -------------------------
```

SPOF Analysis

Postgres-XC Single Point of Failure

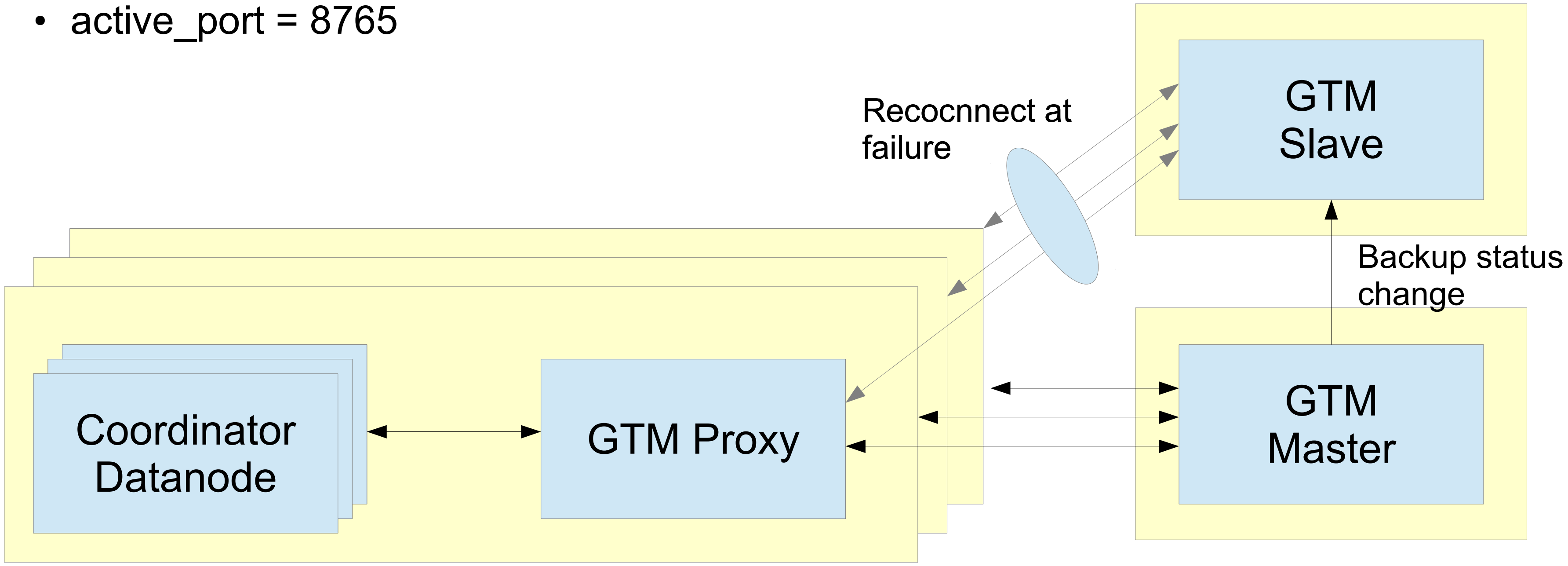
- GTM
  - Obviously SPOF
- GTM-Proxy
  - No persistent data hold
  - Just restart when fail
- Coordinator
  - Every coordinator is essentially a copy
  - When fails, other coordinators work
- Datanode
  - SPOF for sharded table

Backup for SPOF component

- GTM
  - Specific backup for GTM (GTM Standby)
    - Most information are kept on-memory
      - Open TXNs
        - Only the next GXID is needed to restart whole cluster, kept on disk.
    - Copies every internal status change to the backup
      - Similar to the log shipping in PostgreSQL
    - Can promote to the master
      - GTM-Proxy help this failover
- Datanode
  - Need backup
  - Can use PostgreSQL's means
    - Log Shipping
    - Shared disk
- Coordinator
  - Not critical but may want to have backups
  - Can use similar means as Datanodes.

GTM Slave (GTM Standby)

- Same binary to GTM
  - Backs up everything on the fly.
  - Can promote to the master (gtm_ctl promote)
  - Configure using gtm.conf
    - startup = ACT|STANDBY
    - active_host = 'active_gtm_host'
    - active_port = 8765



Sept. 18th, 2013

Datanodes

- Almost all the techniques for PostgreSQL backup/failover are available
  - Streaming replication
  - Shared disk re-mount
- Subject to coordinators
  - Coordinators should reconfigure failed datanode at failover
  - Coordinators should clean connections to failed datanode before reconfiguration
- GTM
  - Reconnect to (new) local GTM proxy

Coordinators

- Only catalog is stored
  - Very stable and static
  - All the coordinators are essentially the same copy
- Datanode HA technique can be applied
  - Streaming replication
  - Shared disk remount
- One more option at a failure
  - No failover
  - Remaining coordinators will take care of TXNs
  - Failed coordinator can be restored offline
    - Backup/restore
    - Copy catalogue from a remaining coordinator

XC vs. O*R*

Feature	Postgres-XC	O____ R__
Background Databsae	PostgreSQL	O_____
Architecture	Shared Nothing	Shared Everything
Number of Servers	Experience: 10 Maybe 20 or more	?? (Most deployments are two server configuration)
Hardware Requirement	None	Shared Disk
Read Scale	Yes	Yes
Write Scale	Yes	Depends (Application level partitioning)
Storage Failure	Limited impact Component failover Cluster keeps running	Whole cluster fails Cluster-wide failover
Server Failure	Affected components needs failover Others keep running	Remaining servers continues service

What's Next?

Current and Future Roadmap (1)

- V1.1 (July 2013)
  - Dynamic component addition/removal
  - Table redistribution
  - Trigger
  - Returning
  - Cursor
  - Planner improvement
  - PostgreSQL 9.2.4 merge
  - Many others …
- V1.2 (Dec. 2013)
  - More robust JDBC
  - Planner improvement
  - PostgreSQL 9.3 merge
  - GUI and HA tools (external project)

Current and Future Roadmap (2)

- V1.3 or later …
  - Concurrent Table Redistribution
  - Flexible function pushdown
  - Coordinator/Datanode integration
  - Single XC node as a standalone database
  - Savepoint
  - Repeatable read/SSI
  - GTM Proxy as a coordinator backend
  - Continue to merge with upcoming PostgreSQL
  - Many more ...

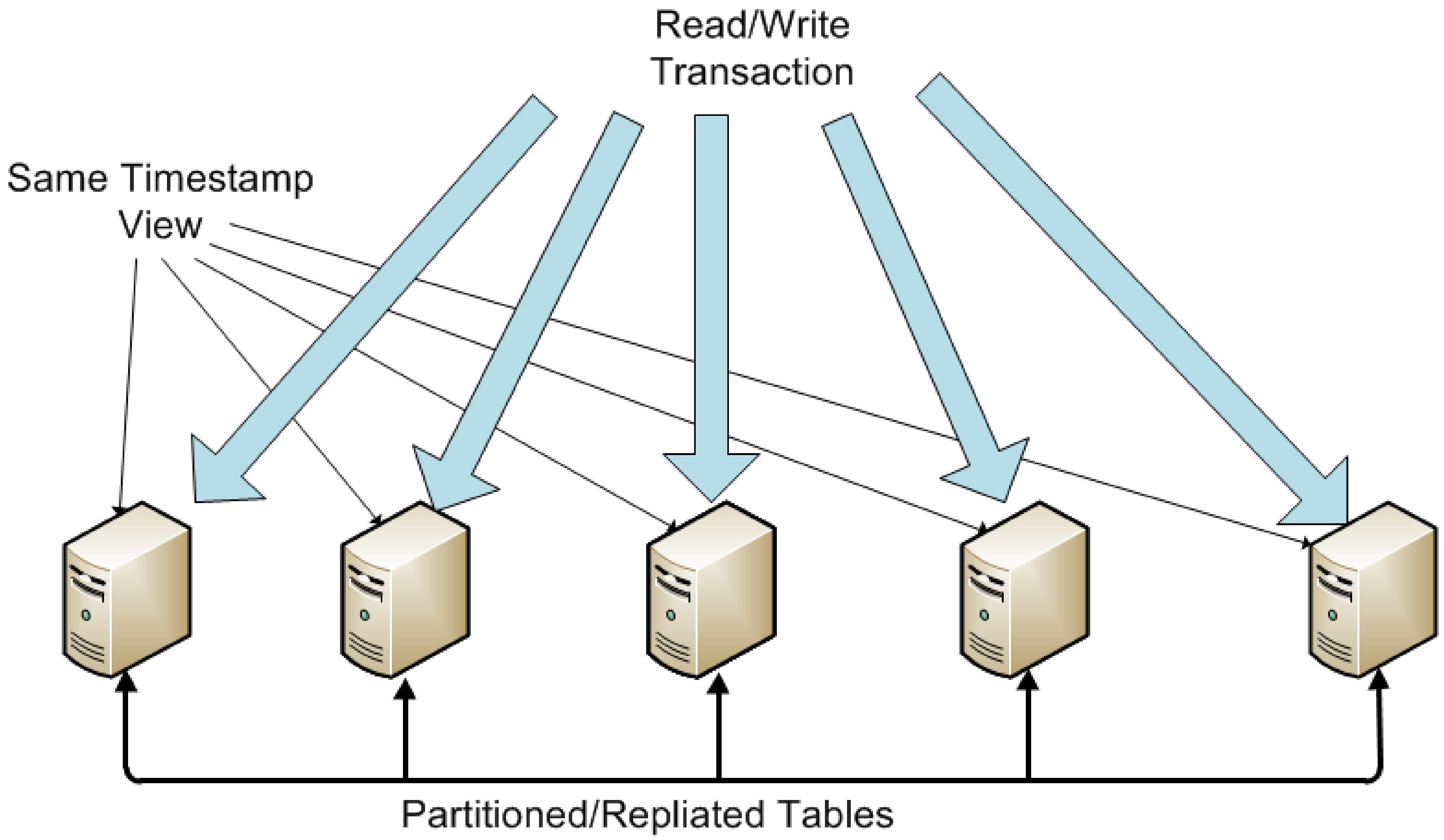
Thank you very much!!

Postgres-XC Short Review

What is Postgres-XC

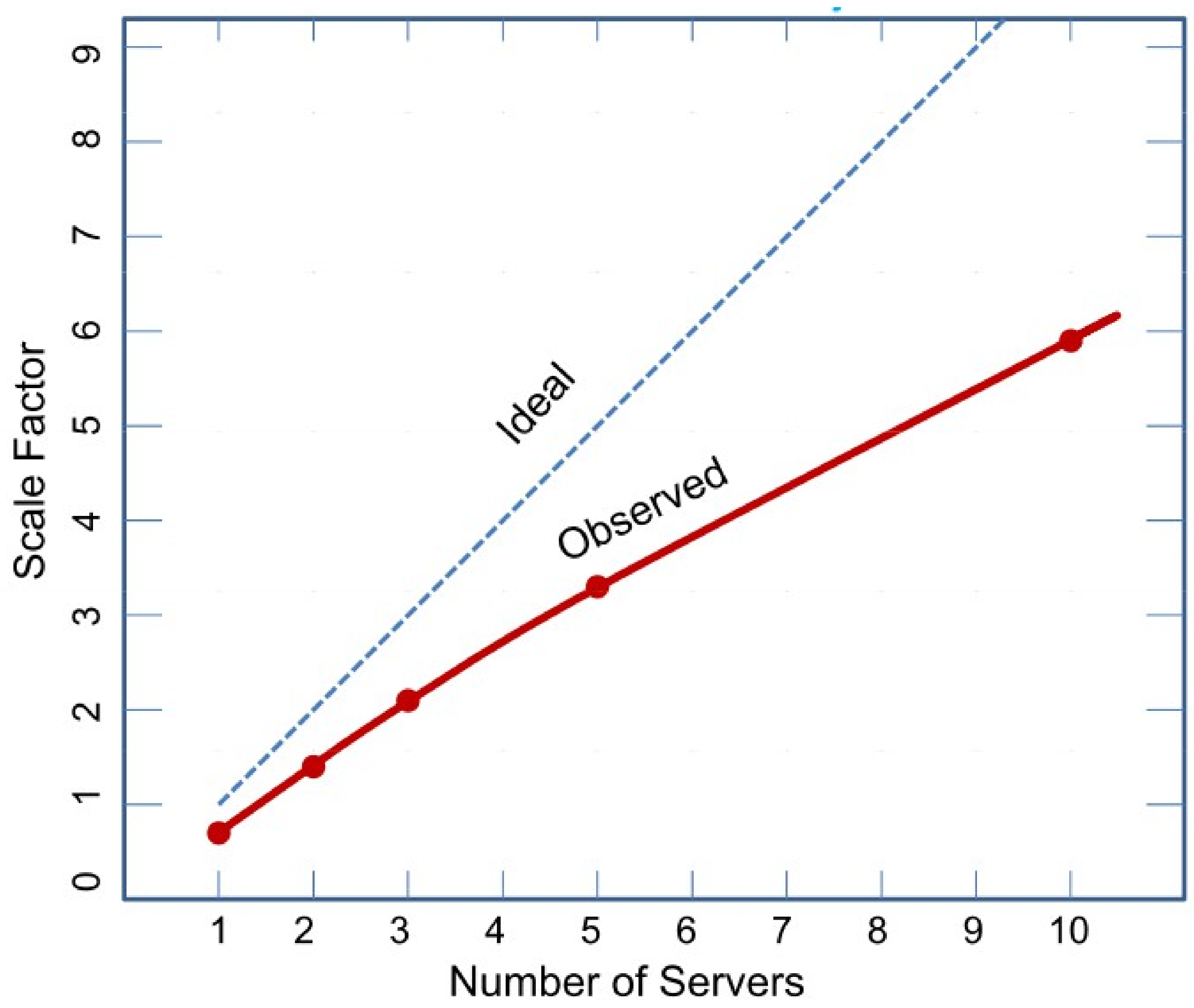
- Symmetric PostgerSQL cluster
  - No master/slave replication
  - No read-only clusters
  - Every node can issue both read/write
  - Every node provides single consistent database view
  - Transparent transaction management
- Not just a replication
  - Each table can be replicated/distributed by sharding
  - Parallel transaction/query execution
    - So both read/write scalability

Postgres-XC Symmetric Cluster



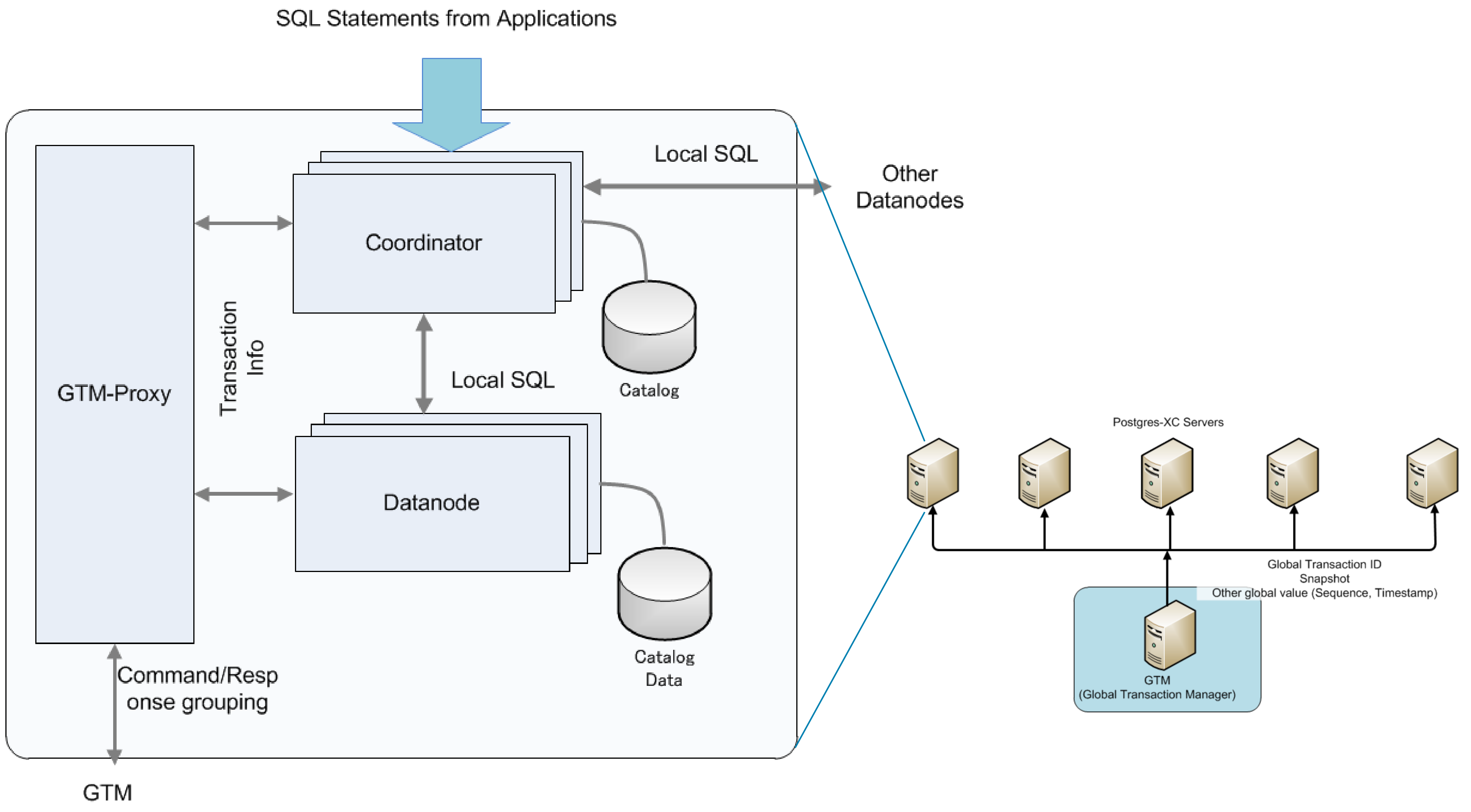
Sept. 18th, 2013

Scalability



DBT-1 (Rev)

Architecture and Configuration



Sept. 18th, 2013

Outline of Components

- GTM (Global Transaction Manager)
  - Distributed MVCC
    - Provide global transaction ID (GXID) to all the transactions
    - Provide global snapshot to all the transactions
  - Sequence
- GTM_Proxy
  - Group communications to GTM and reduce amount of GTM network workload
- Coordinator
  - Handles incoming SQL statements
    - Parse, plan, conduct execution in datanodes and the coordinator.
    - Integrate local results from each datanode involved.
- Datanode
  - Actual data storage
    - Almost vanilla PostgreSQL

} Share the binary

Sept. 18th, 2013

Flexible Configuration of Comonents

- Each coordinator/datanode can be configured in any servers, same or different, as log as
  - Each component does not share the following set of resources
    - Listening IP addresses
    - Listening port
    - Work Directories
- For simplicity and better workload balance, the following is advised:
  - Have separate GTM server
  - Each of others should have
    - One GTM proxy (for network workload improvement)
    - One Coordinator
      - Some transactions may benefit from data located at local datanode (preferred node)
    - One Datanode
      - Automatic workload balance between coordinator and datanode

How XC Scales both Read/Write

- Transaction Tables → Sharding
  - Only one write
  - Parallel writes in datanodes
- Master Tables → Replication
  - Relatively static: Not significant many-writes overhead
  - Local join with transaction tables → Most join operation can be done locally in datanodes