

Postgres-XC
PG session #3

Michael PAQUIER
Paris, 2012/02/02



Agenda

- Self-introduction
- Highlights of Postgres-XC
- Core architecture overview
- Performance
- High-availability
- Release status



Self-introduction

- Michael Paquier, 27 years old.
- Based in Tokyo since 2009
- Working for NTT Data Intellilink
 - _ 500 employees
 - _ Website <http://www.intellilink.co.jp>
 - _ Company of NTT Data Group: 55,000 employees
- Working on DB system support mainly PostgreSQL
- PostgreSQL community member
 - _ pgbench shell-related features of 9.0
 - _ 2PC studies
- Core developer of Postgres-XC
- Other information:
 - _ Email: michael.paquier@gmail.com
 - _ Twitter: [@michaelpq](https://twitter.com/michaelpq)
 - _ Blog: <http://michael.otacoo.com>



Highlights



Highlights - Postgres-XC

- Cluster software focused on write-scalability
- Based on PostgreSQL
 - world's most advanced open source database
 - PostgreSQL license
- Same client APIs as PostgreSQL
 - Ease of application migration from existing PostgreSQL deployment
 - Same drivers, same front end, same SQL queries
- Licensing
 - PostgreSQL license (more or less BSD)
 - Free to use, modify and redistribute for commercial purposes



History

- Started through a collaboration between EnterpriseDB and NTT Open Source Software Center in January 2009
- Goal to build a PostgreSQL based clustering solution which can serve as an alternative to Oracle RAC
- Development is community-based, with resources gathered from NTT and EnterpriseDB
- Licensing terms changed from GPL to PostgreSQL license (same as Postgres) in 2011



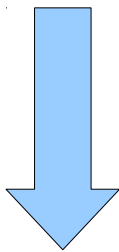
Core architecture



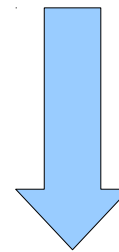
About PostgreSQL 9.1

- Streaming replication and HOT-Standby

Read/Write possible



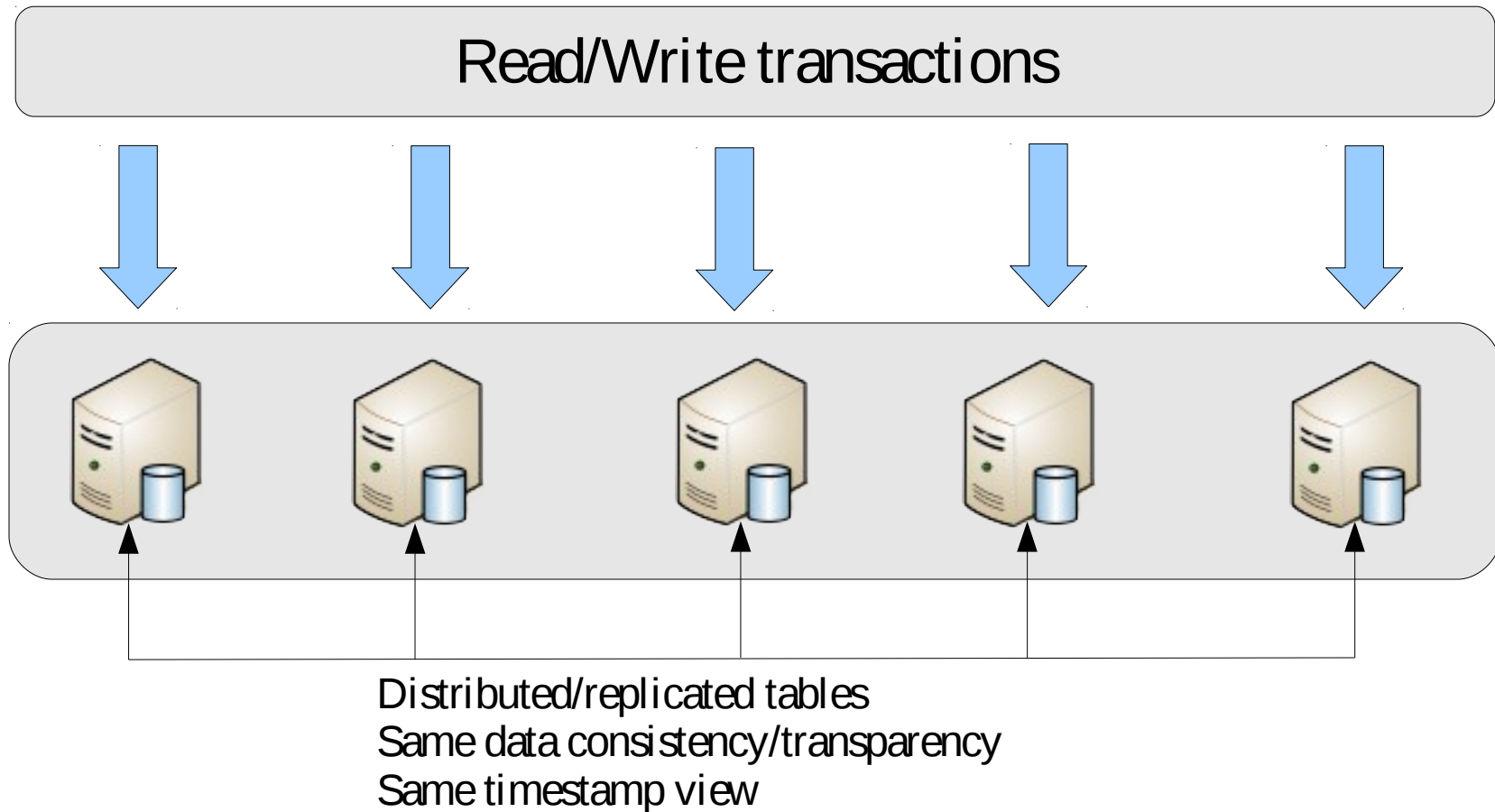
READ only



Log shipping

- Asynchronous mode => timestamp view not consistent
- Synchronous mode => timestamp view consistent

And Postgres-XC itself?



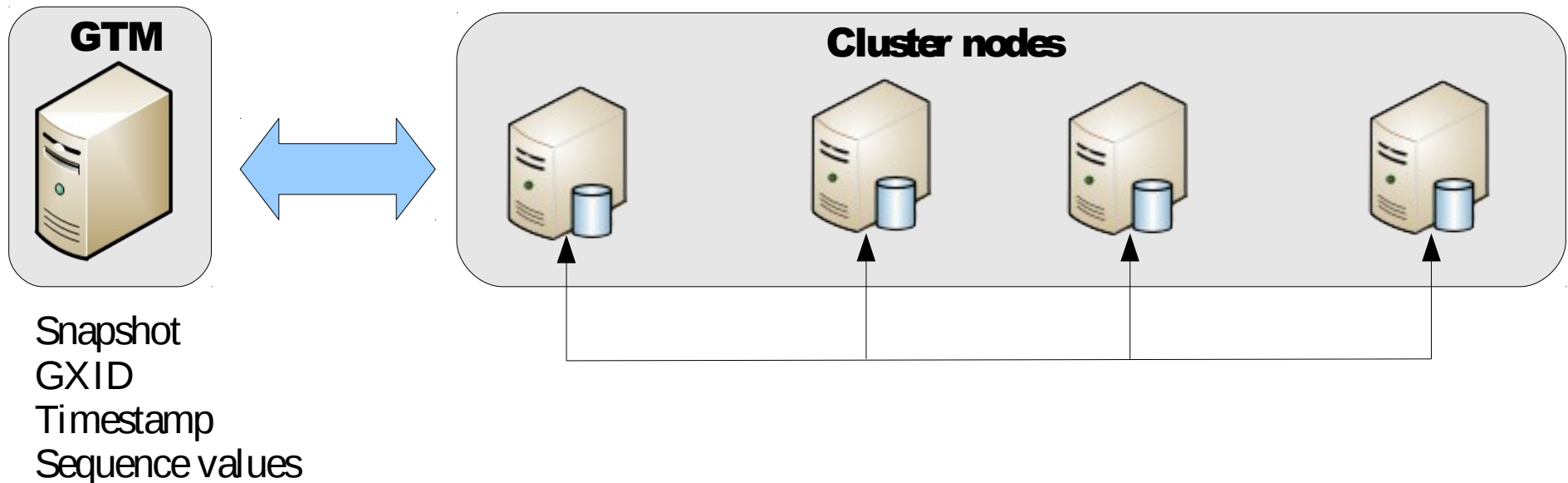
In short

- Symetric cluster of PostgreSQL
 - No Slave and no Master
 - Read and write scalability
 - Transparent Transaction Management
 - Shared-nothing structure
 - 3 types of nodes: GTM, Coordinator, Datanode



Node types – GTM (1)

- Designed for transparency
- Feeding of MVCC-related data: transaction ID, snapshot
- Cluster follows GTM timeline: timestamp, sequence



Node types - Coordinator (2)

- Point of contact for the application/client
- Management of remote node data
 - Parse and partially plan the statements
 - Determine the data to be fetched from remote nodes at planning or execution
 - Fetch the required data by issuing queries to dedicated Datanodes
 - Combine and process the data to evaluate the results of the query (if needed)
 - Pass the results to the applications
- Manages two-phase commit
- Stores catalog data: cluster-related information
- Needs and manages space for materializing results from remote nodes
- Binary based on the latest PostgreSQL release

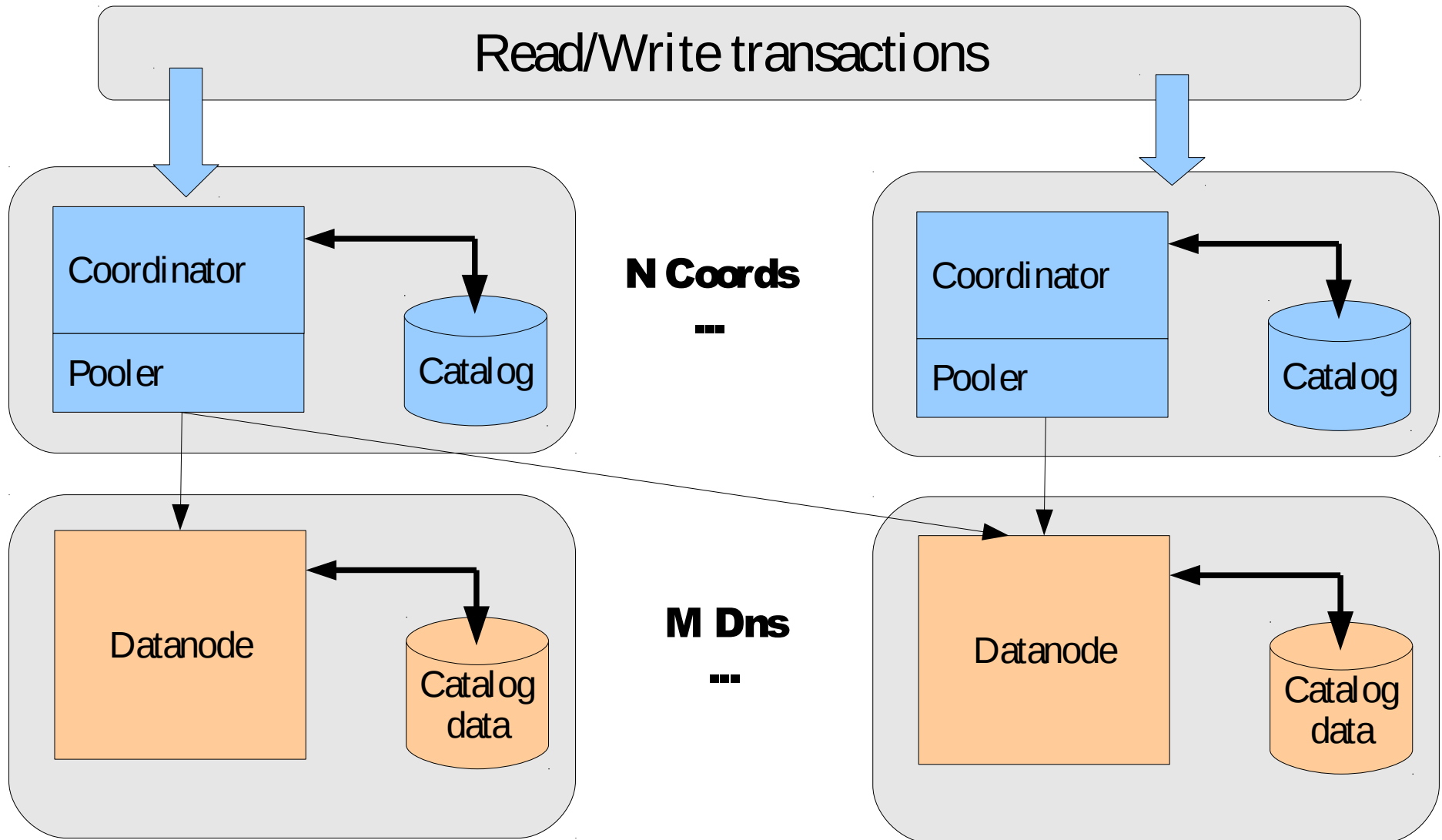


Node types - Datanode (3)

- More or less a PostgreSQL instance (remote node)
- Stores tables and catalogs
- Executes the queries from client Coordinator and returns results to it
- Data nodes can be made fault tolerant by Hot-Standby and Synchronous Replication technologies available of standard PostgreSQL
- Binary same as Coordinator, based on latest PostgreSQL release



Node types - Global (4)

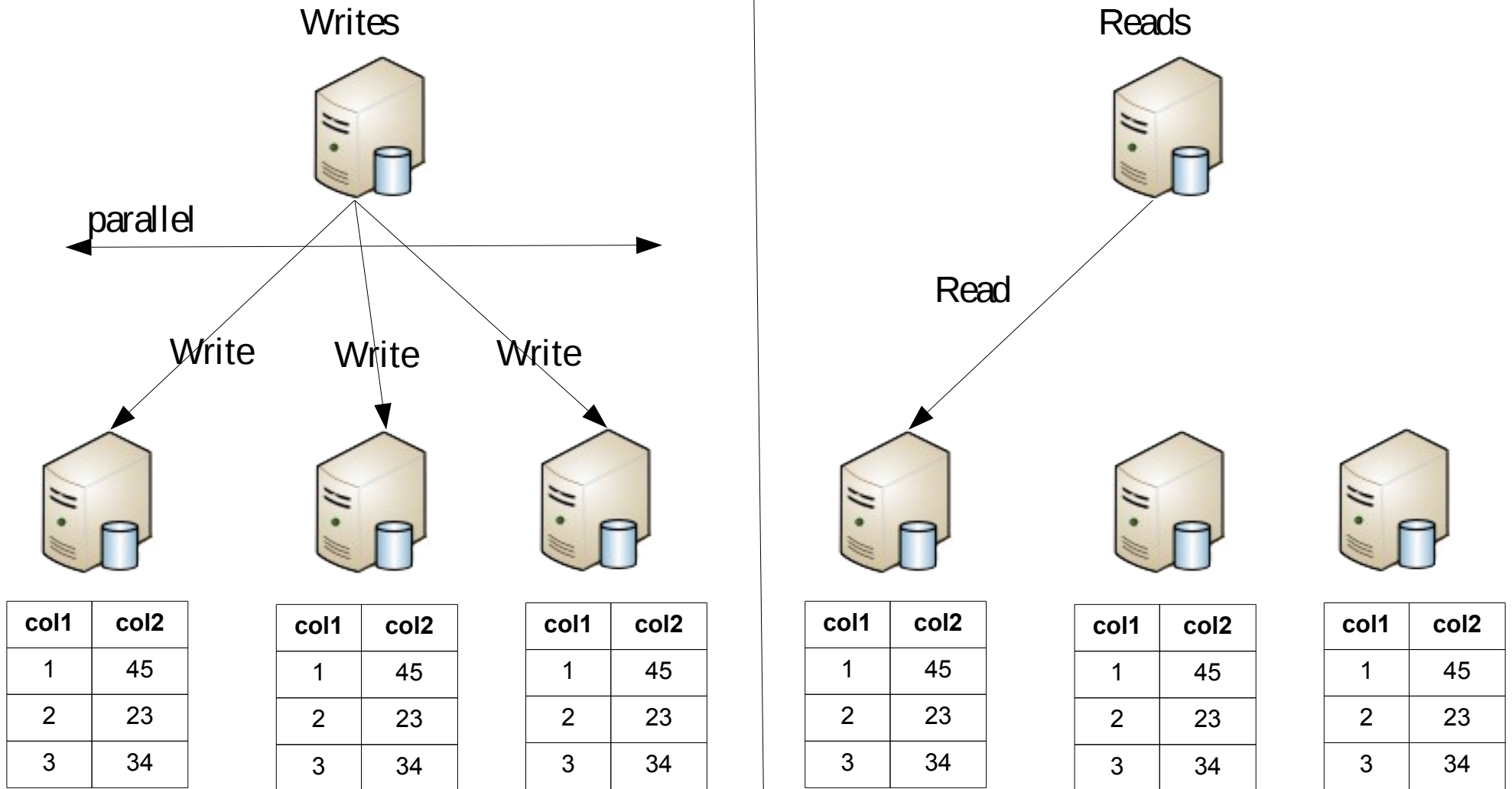


Data management

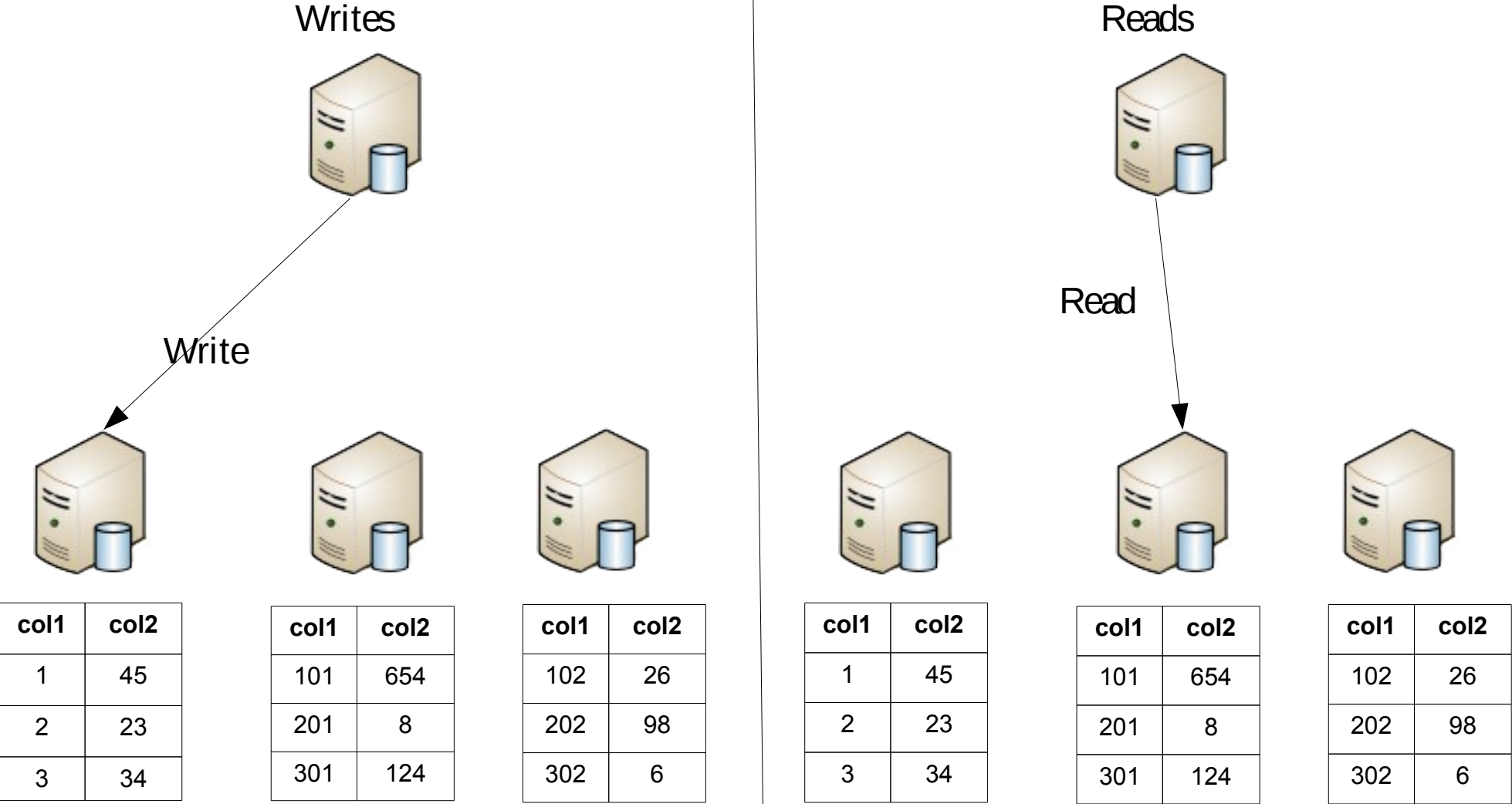
- Table types
 - Replicated table
 - Each row replicated to Datanodes
 - Statement based replication
 - Distributed table
 - Each row of the table is stored on one datanode, decided by one of following strategies
 - Hash
 - Round Robin
 - Modulo
- Managed by SQL extensions (CREATE TABLE)
- Possible to define subset of nodes



Replicated tables



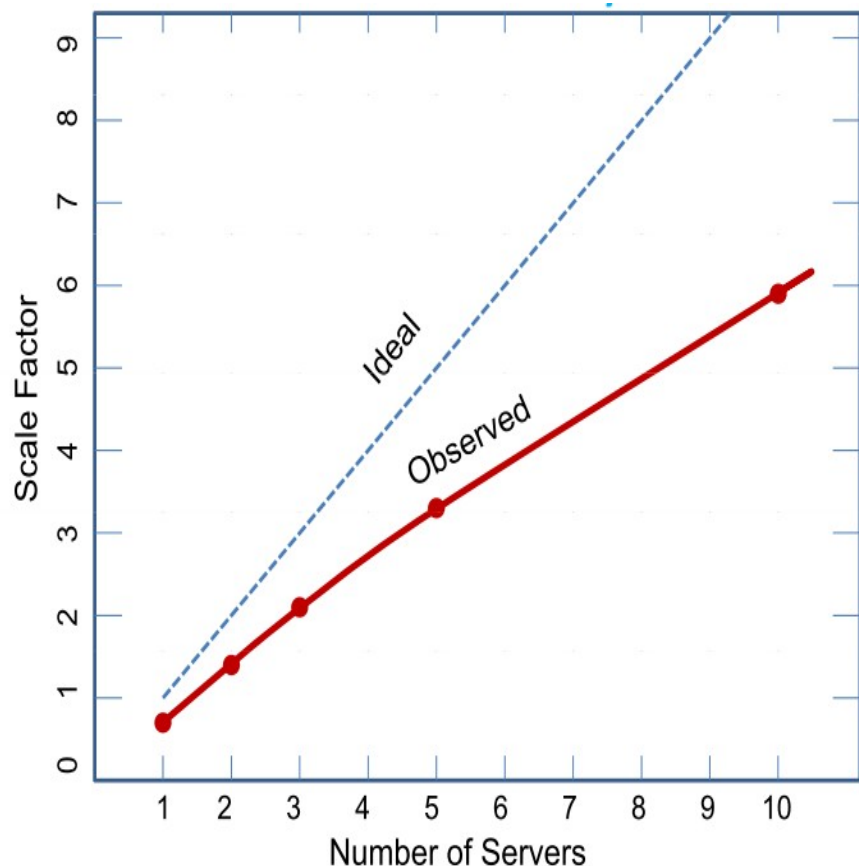
Distributed tables



Performance



Scalability measurements



- Tests done with DBT-1 (TPC-W) benchmark with some minor modification to the schema
- 1 server = 1 coordinator + 1 datanode
- Coordinator is CPU bound
- Datanode is I/O bound
- CPU usage
 - Coordinator 30%
 - Datanode 70%

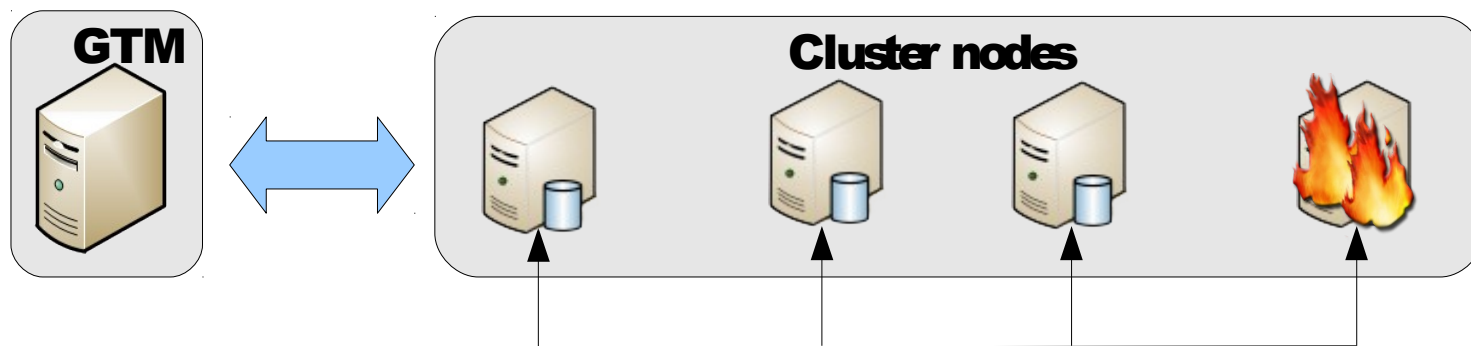


About high-availability

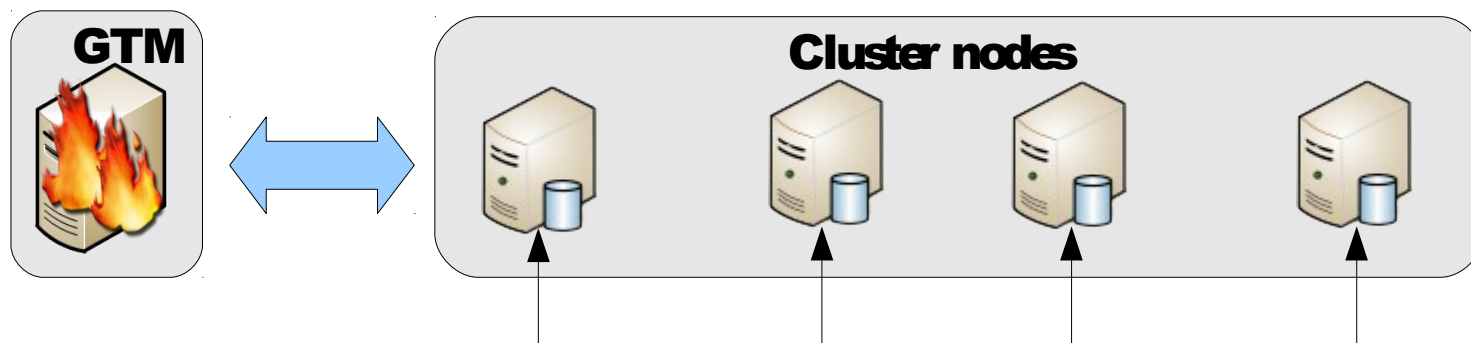


Cluster SPOF problem

- Datanode is a SPOF if it has a portion of distributed table.

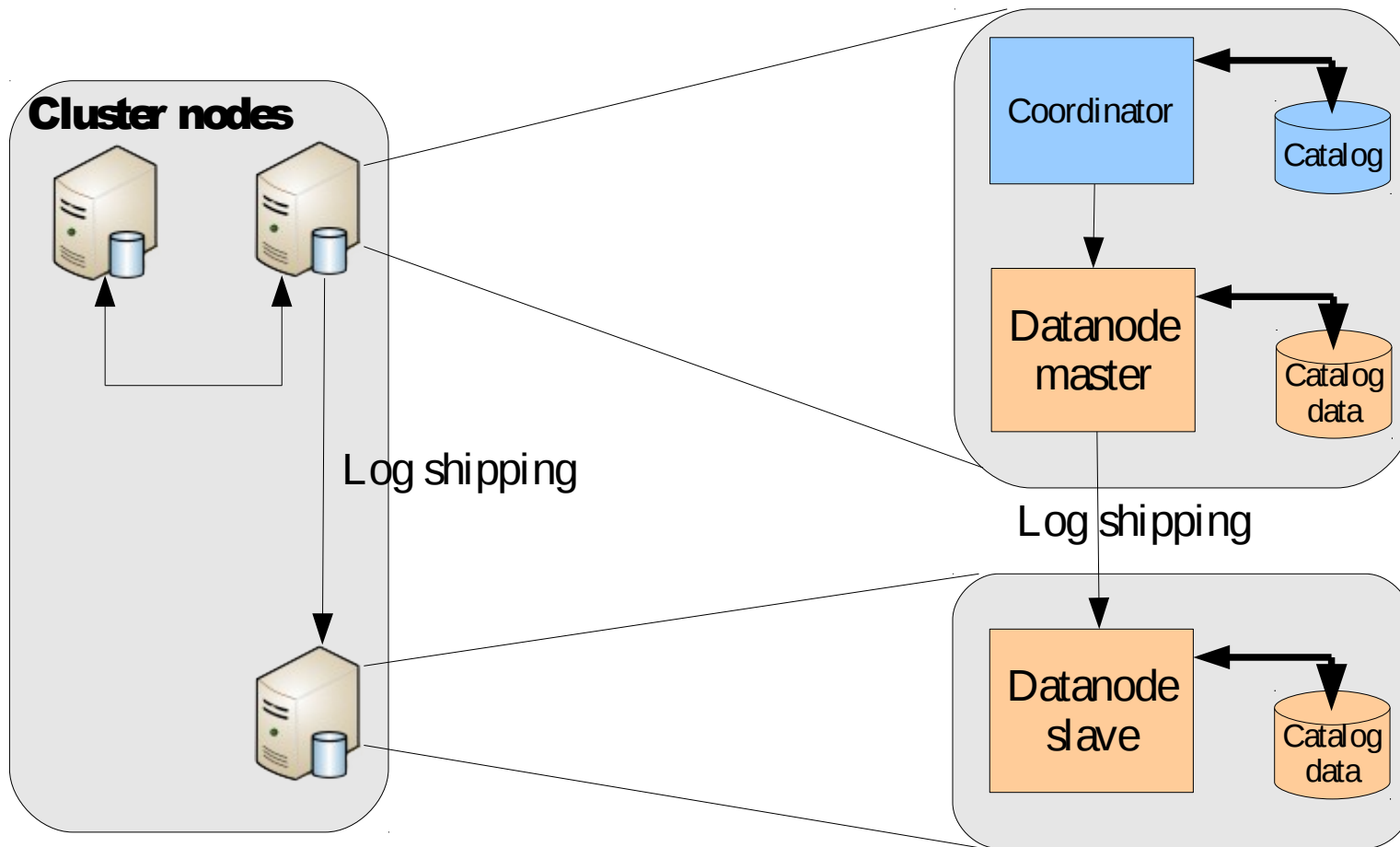


- GTM case



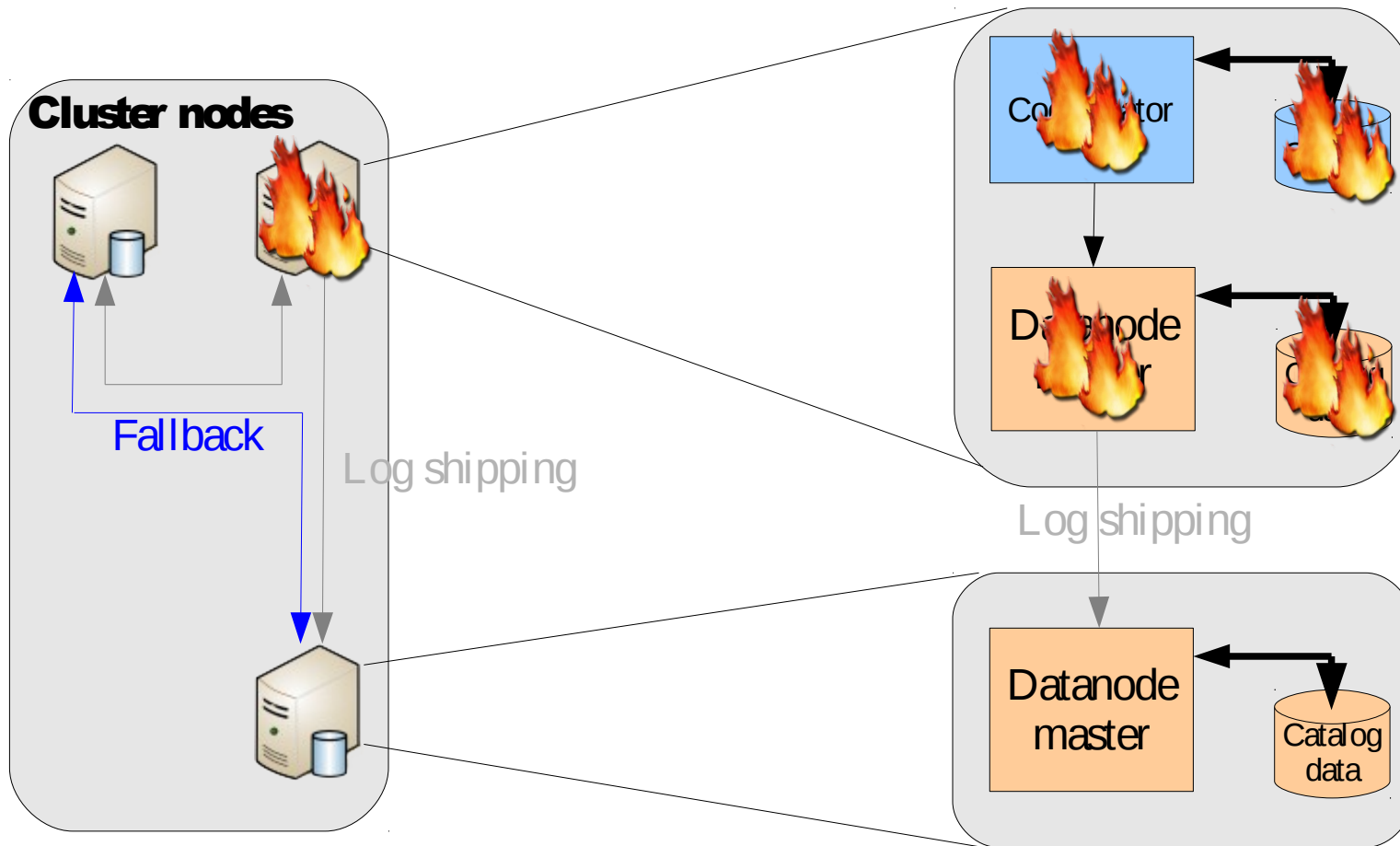
Datanode SPOF resolution (1)

- PostgreSQL 9.1 synchronous stream-rep



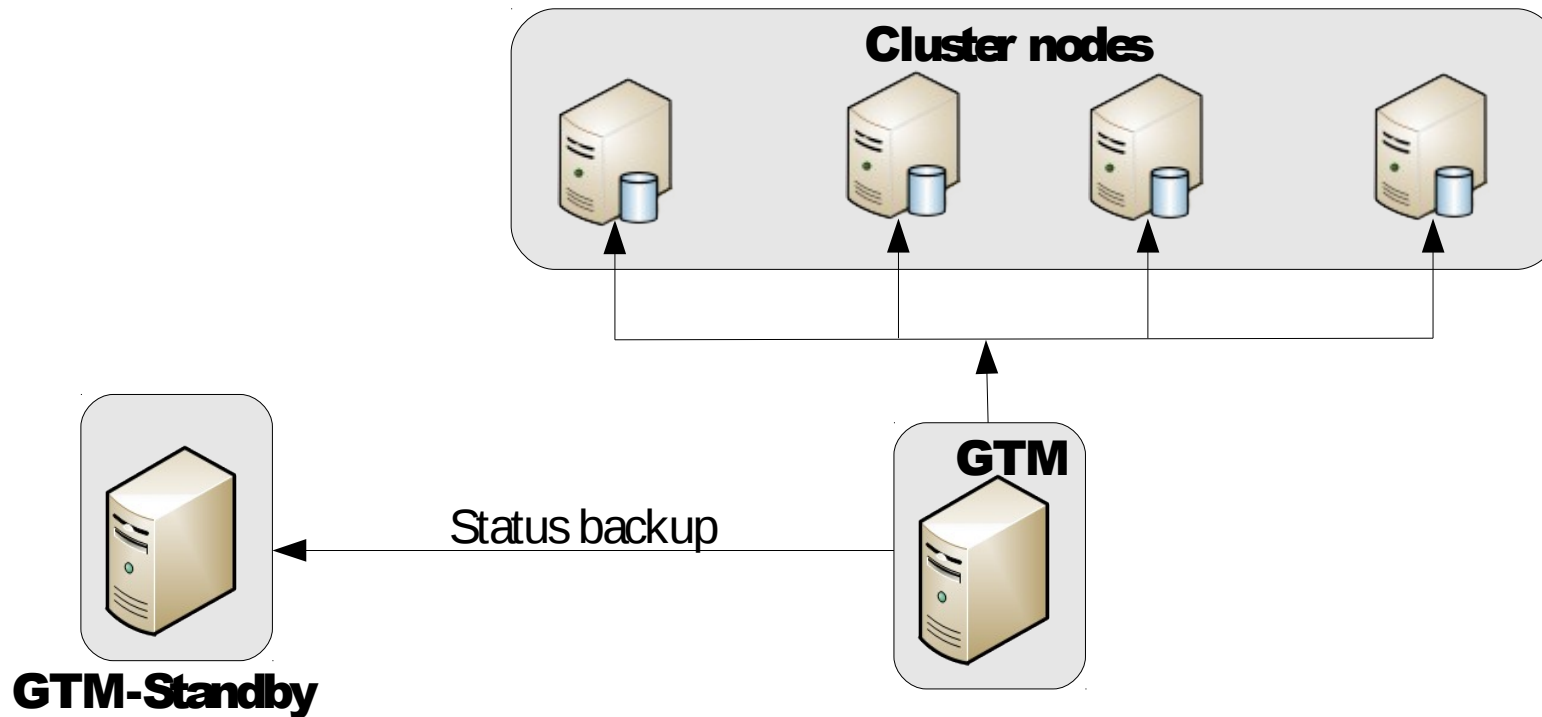
Datanode SPOF resolution (2)

- Fallback slave node



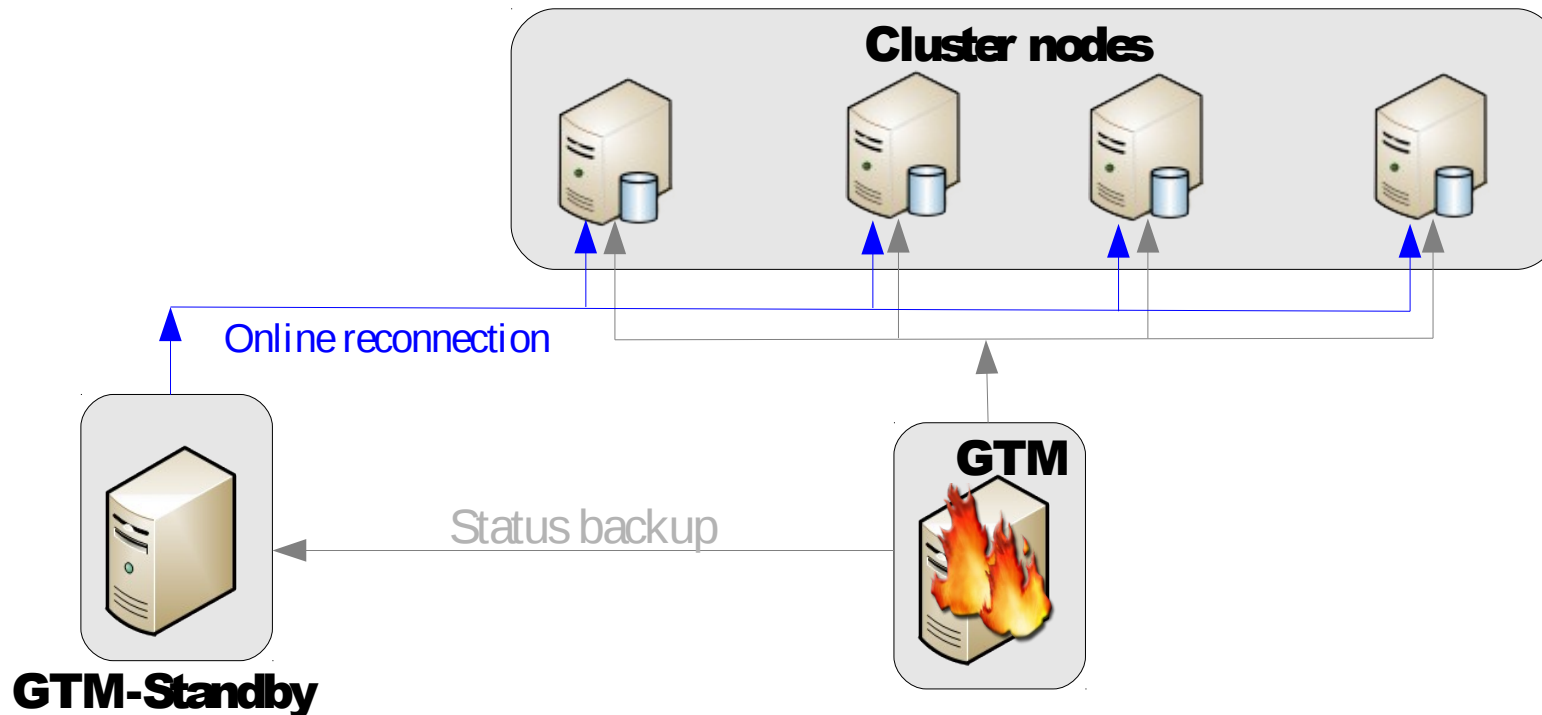
GTM SPOF resolution (1)

- Use of a standby for GTM



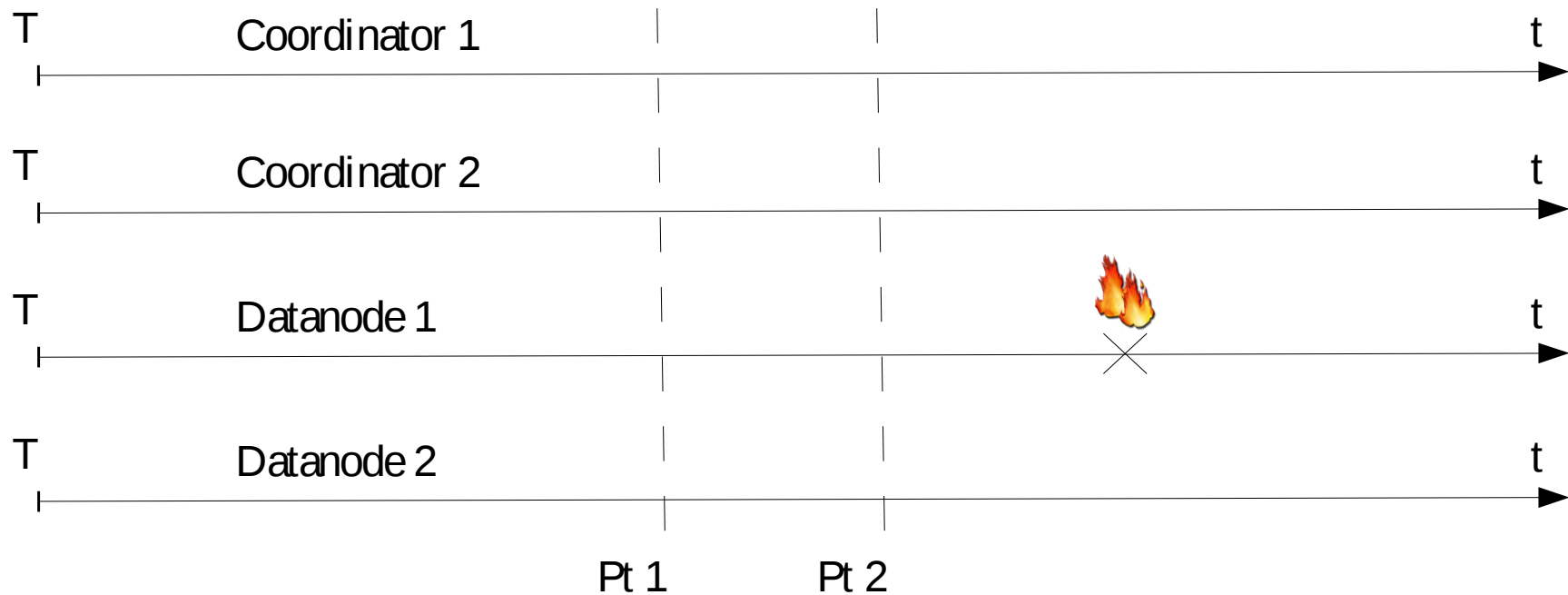
GTM SPOF resolution (2)

- Fall back to standby and reconnect nodes



PITR – requirements (1)

- PITR, Point in-time recovery
 - Rollback the database to a given past state
 - Need consistent points to restore shared-nothing nodes



PITR – requirements (2)

- Transaction status has to be consistent in the cluster
- Each transaction must be either:
 - Committed/Prepared/Aborted/Running on all the involved nodes
 - We must avoid cases where transaction is prepared and committed partially, or prepared and rolled back partially
- Write record in WALs of all the coordinators and datanodes at a moment when all the transaction statuses are consistent.
- External Application can provide such timing as with BARRIER
 - *CREATE BARRIER barrier_id*
- BARRIER:
 - Waits that partially committed or aborted transactions commit (2PC)
 - Blocks all transaction commit when running a barrier
 - Still needs a timeout functionality
- When running PITR, specify `recovery_target_barrier` in `recovery.conf`



Release status

What now and next?



Current functionalities

- Up to 0.9.7 (current release of January 2012)
 - Based on PostgreSQL 9.1
 - 90%-92% of SQL
 - Major DDL/DML (TABLE, ROLE, VIEW, SELECT INTO, DEFAULT values)
 - General select support: support extension
 - HAVING, GROUP BY, ORDER BY, LIMIT, OFFSET, aggregate, window function, etc.
 - SQL-based cluster setting
 - Relation-size functions



About Postgres-XC 1.0

- Release on April 2012 (plan)
- Tablespace
- Triggers
- SERIAL
- Cluster bootstrap
- SELECT FOR UPDATE



After 1.0

- Node addition/removal
 - Move tuples from a node to another node
 - Ex: update of a distribution column
- Online server removal/addition
- Connection balancing between master and slave Datanodes for read transactions.
- SQL/MED, Foreign data wrapper (FDW) integration
- Installation, configuration, operation
- Global deadlock detection (global wait-for-graph)



Project resources and contacts

- Project home
 - <http://postgres-xc.sourceforge.net>
- Developer mailing list
 - postgres-xc-developers@lists.sourceforge.net
 - postgres-xc-general@lists.sourceforge.net
- Contacts
 - michael.paquier@gmail.com
 - koichi.szk@gmail.com
- Twitter: [@michaelpq](https://twitter.com/michaelpq)
- Blog: <http://michael.otacoo.com>

Sponsored and
supported by:

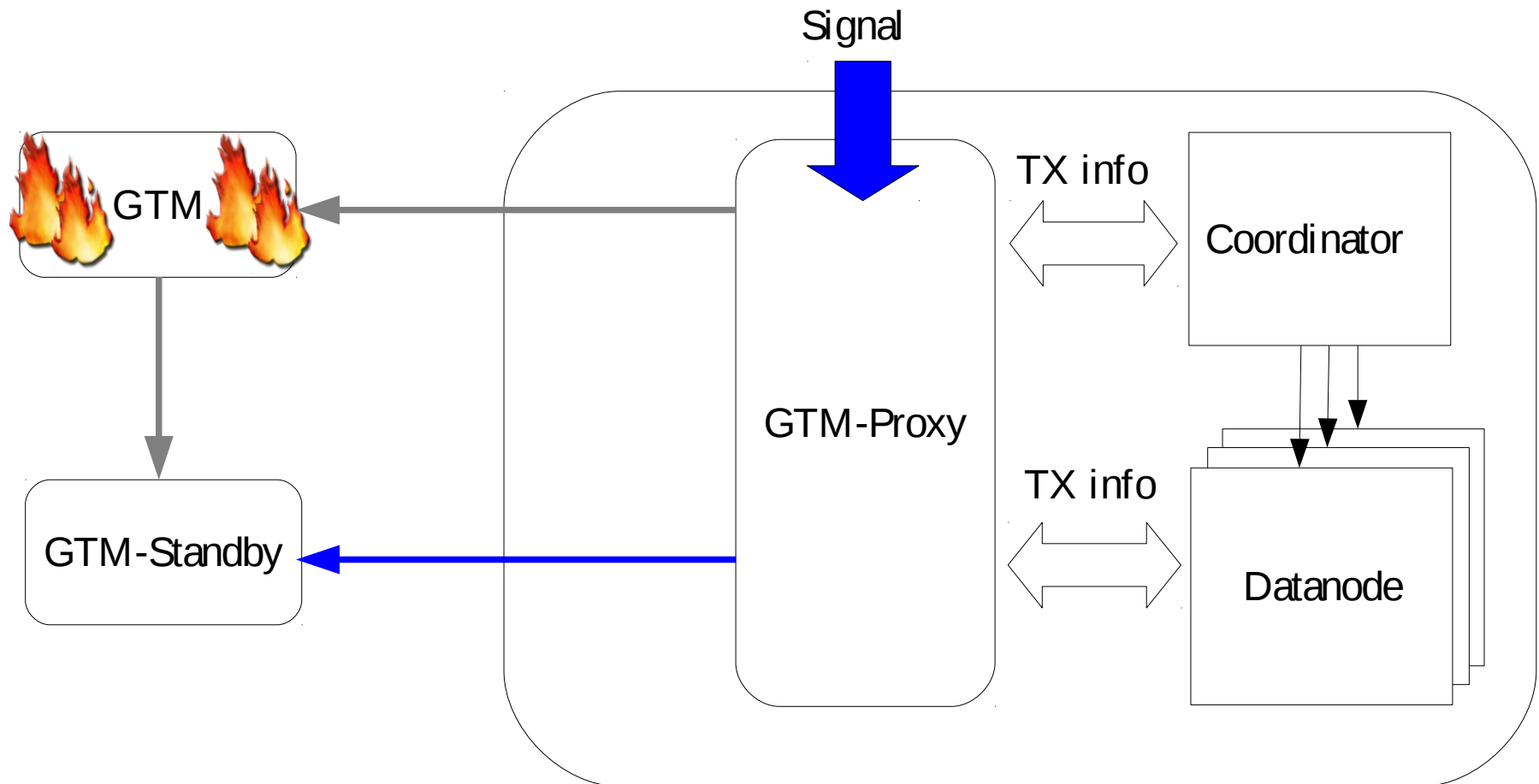


Thanks for your attention.
Questions?

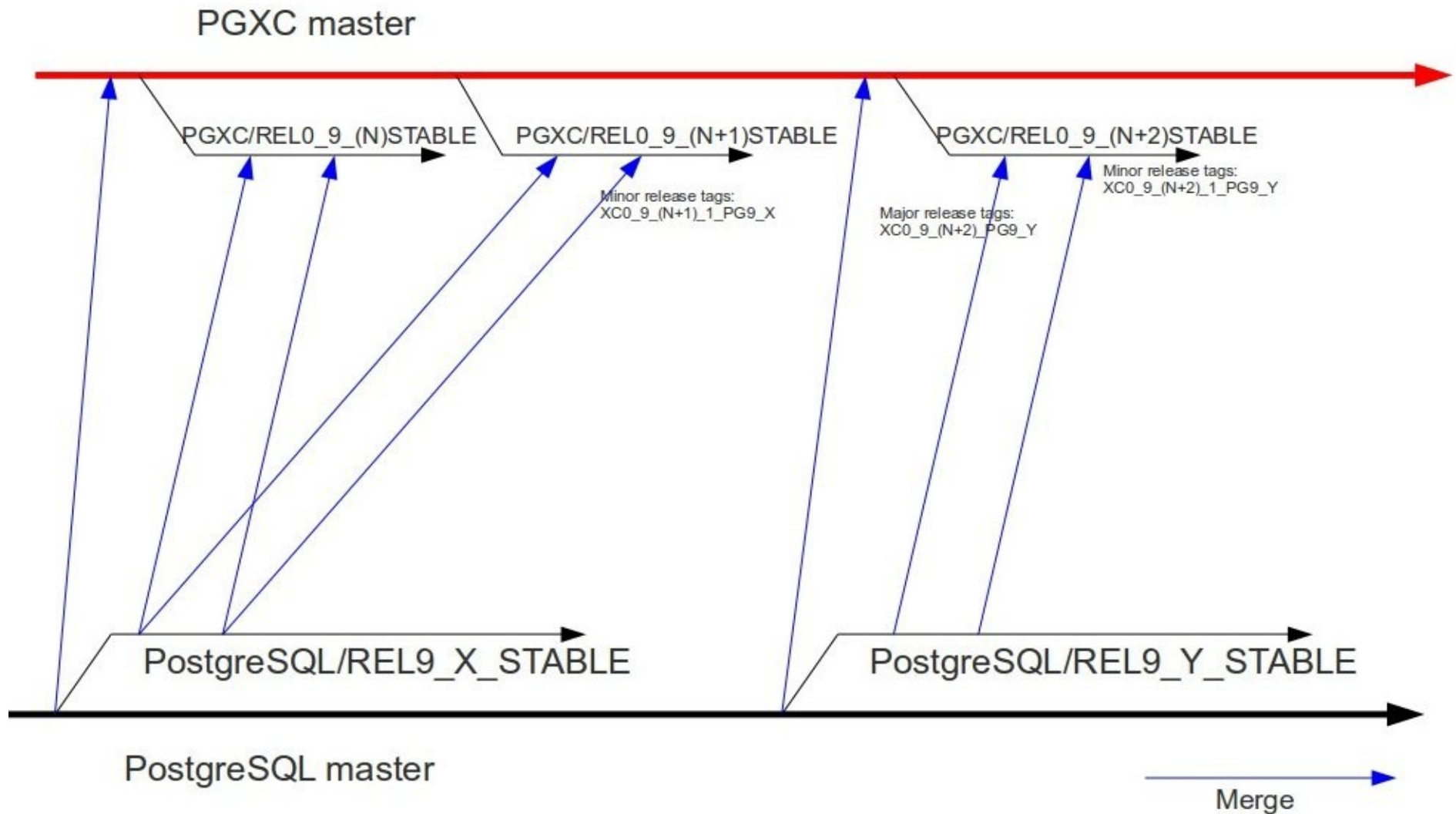


GTM Proxy reconnection

- Signal GTM Proxy and reconnect nodes



Release policy

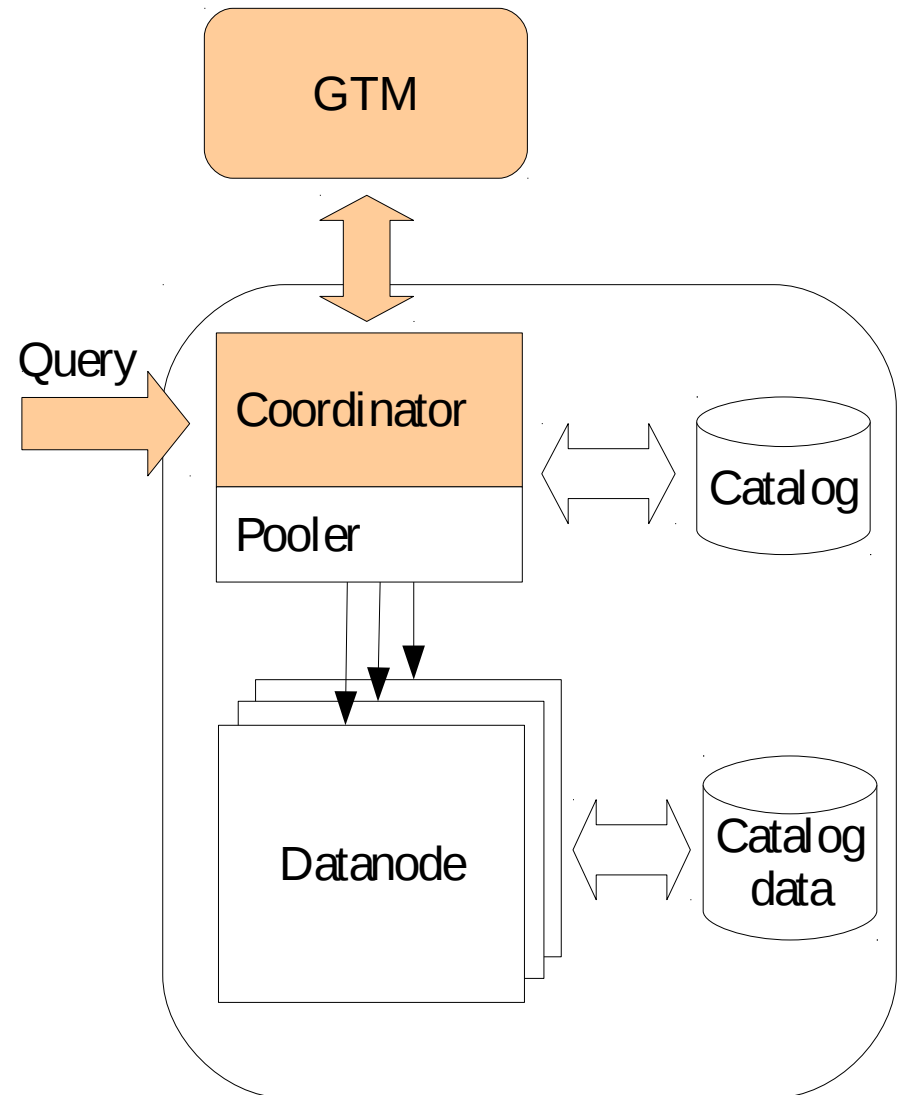


Key algorithm



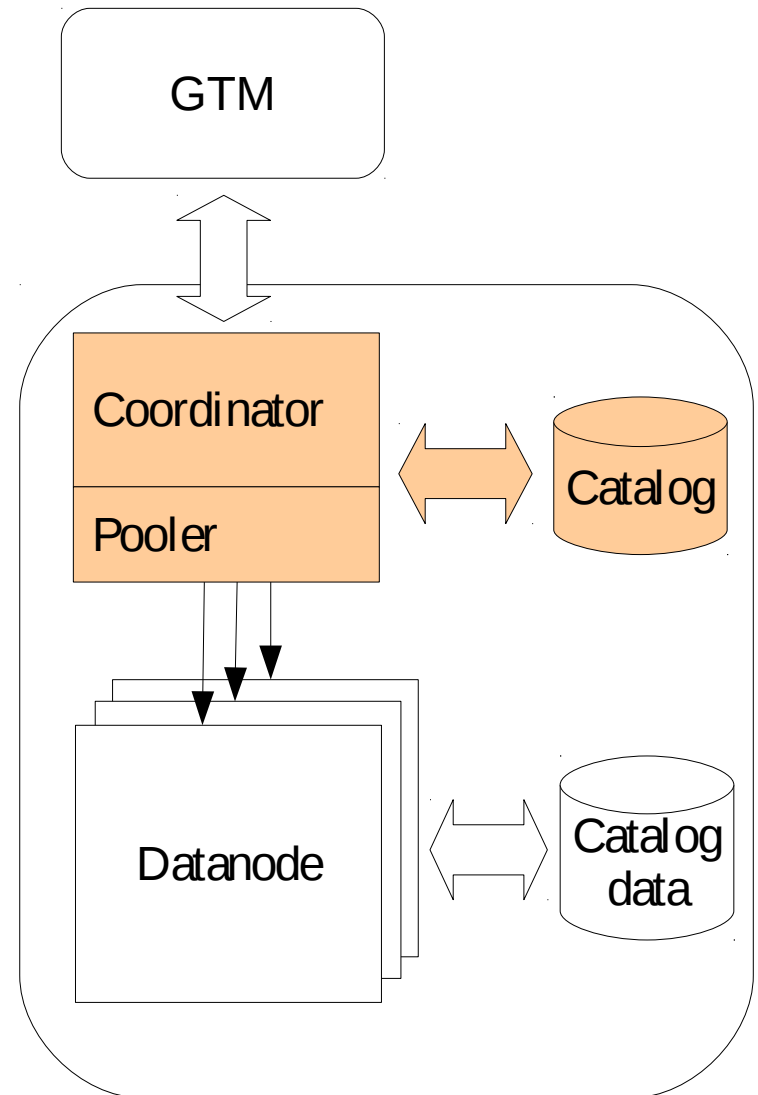
Query algorithm (1)

- Receive query from application
- Get snapshot, GXID and timestamp from GTM



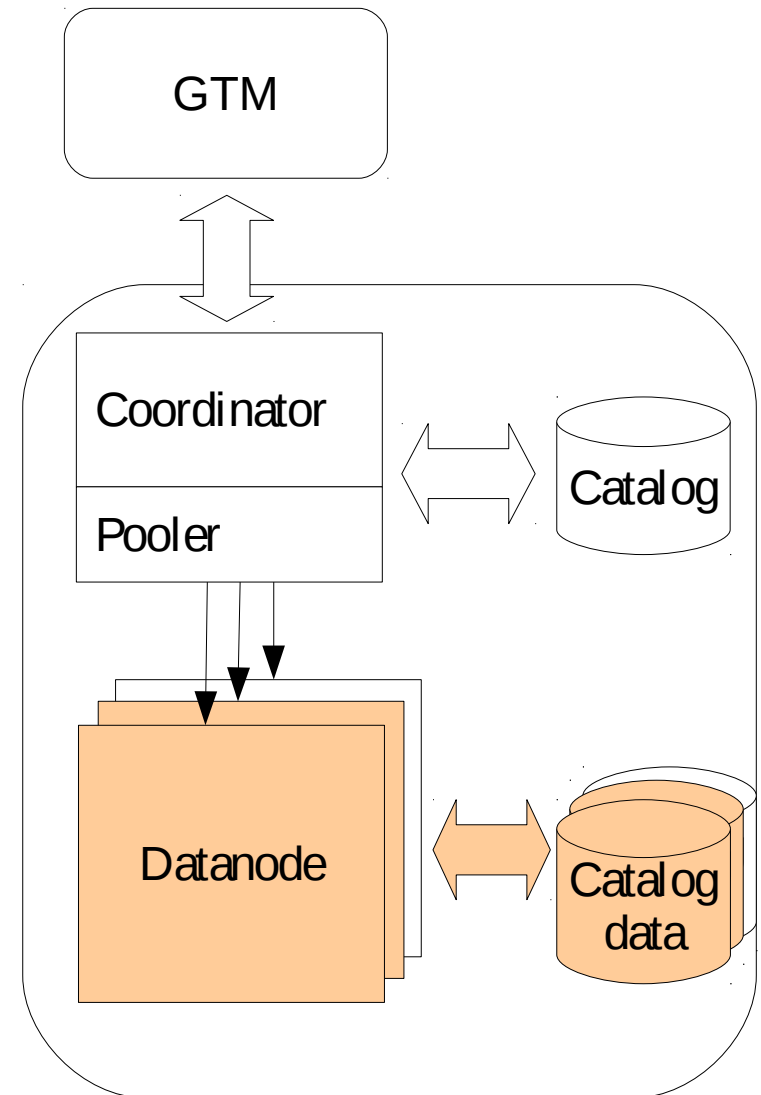
Query algorithm (2)

- Incoming statements: analyzer and rewriter
- Planning: analyze nodes to be involved. Build queries for local nodes (push down if necessary)



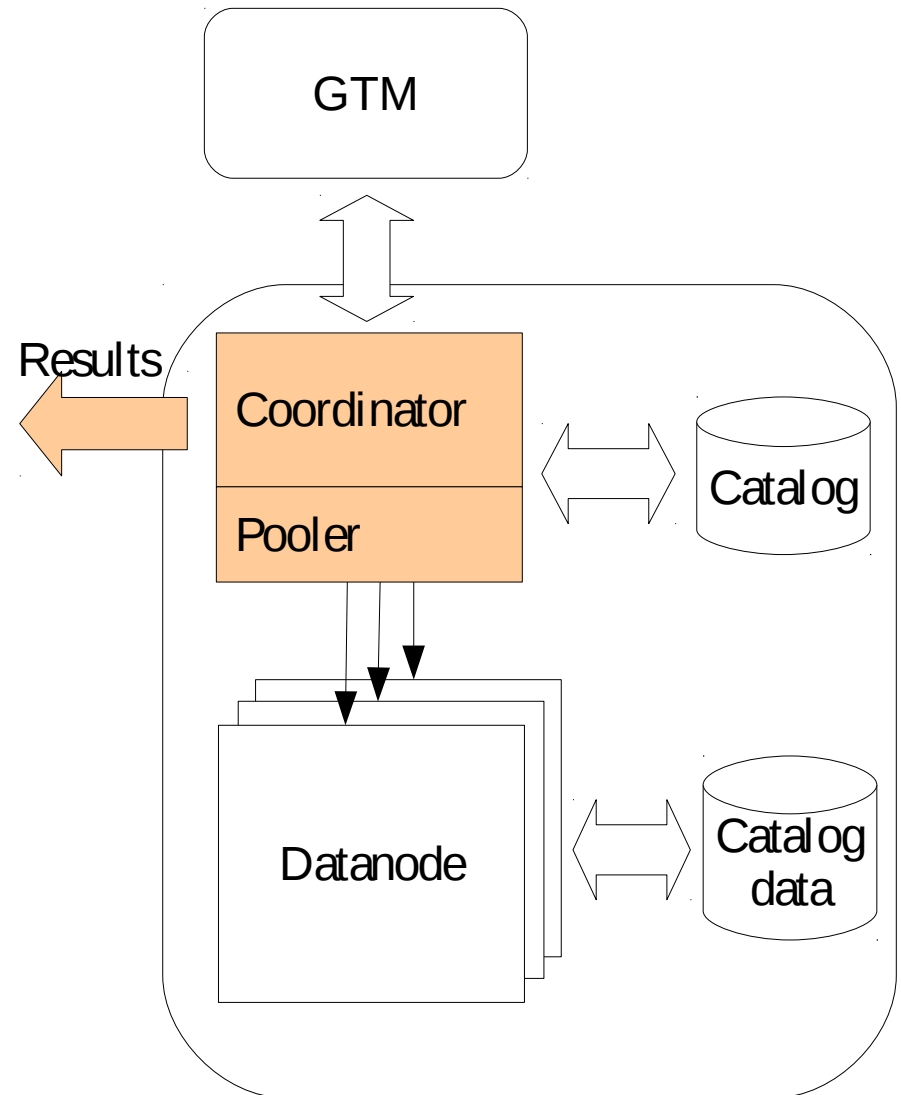
Query algorithm (3)

- Run queries on remote Datanodes and send back results to Coordinator



Query algorithm (4)

- Materialize results if necessary and send back to client



Demonstration

