PostgreSQL Database Replication Options

Darren Johnson
darren@up.hrcoxmail.com
The PGReplication Project

Agenda

Characterizing Replication

Replication Scenarios

PostgreSQL Replication

Postgres-R Concepts

PGReplication Project

Transaction Processing

- **Transaction**- a group of SQL commands whose result will be made visible to the rest of the system as a unit when the transaction commits--or not at all, if the transaction aborts.
- **Transaction Processing Application**- a collection of transaction programs designed to automate a given business activity.

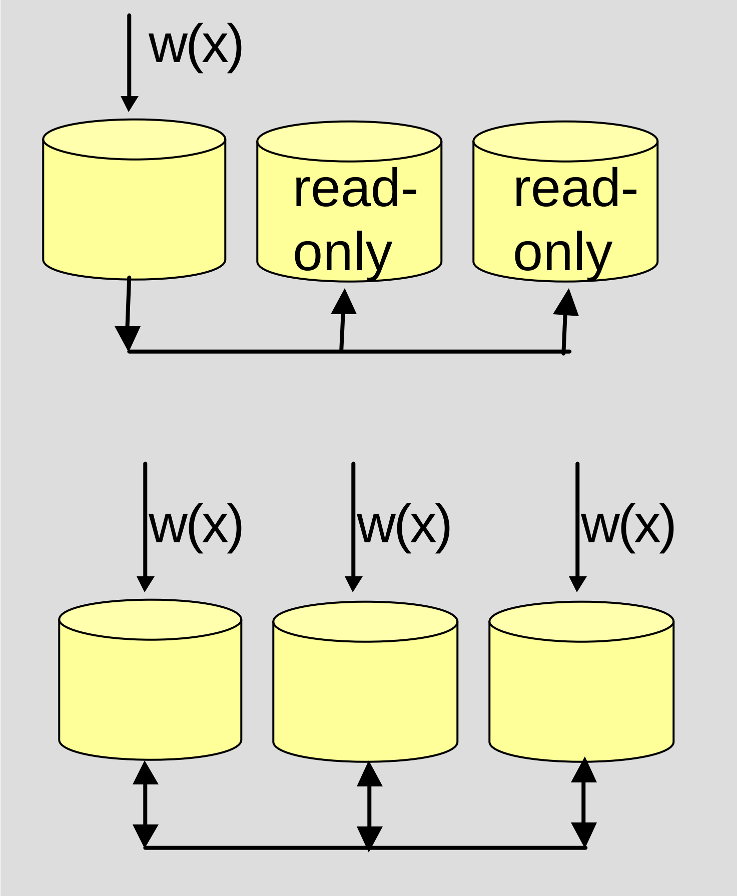
Critical Properties of a Transaction ACID

- **Atomicity**-successful transaction commit; failed transaction abort.
- **Consistency**-each transaction is programmed to preserve database consistency.
- **Isolation**-each transaction is executed as if it were running alone.
- **Durability**-the result of a committed transaction is guaranteed to be on stable storage.

Updating the Database

- **Read Only** (Primary copy / master - slave)

- **Peer to Peer** (Update everywhere / multi-master)



Propagating Updates

- **Asynchronous** (Lazy / Store and Forward) - Post commit sends information to all other systems involved in the distribution.
  - Trade Off: Data synchronization and conflict resolution
- **Synchronous** (Eager) – Pre commit sends information to all other systems in the distribution and verifies a commit or roll back on each transaction for the entire distribution.
  - Trade Off: Performance and scalability

More Characteristics

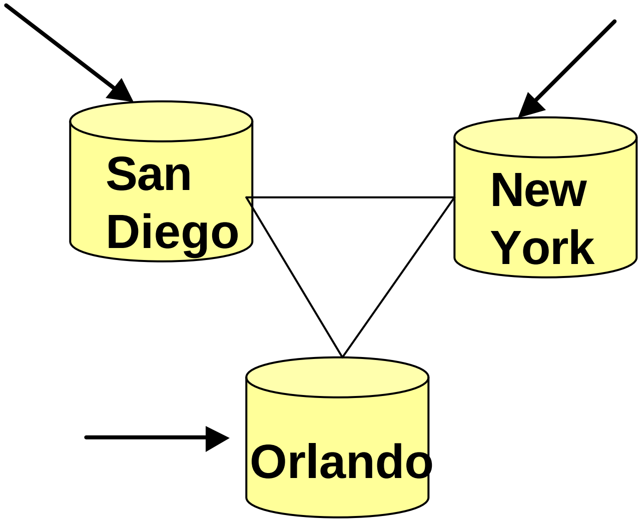
- Propagated as – SQL vs. Parsed (tuple)
- Event driven – Logs vs. Triggers
- What to replicate – Partial vs. Full
- Architecture - Embedded vs. External
- Where to replicate – Pre vs. Post

Scenario 1 (Hot Fail Over)

- The ability to fail a database from a primary standalone server to a secondary server.
  - Usually done locally
  - Hardware/Software solutions
  - Multi-ported RAID
  - Heartbeat
  - WAL services must exist

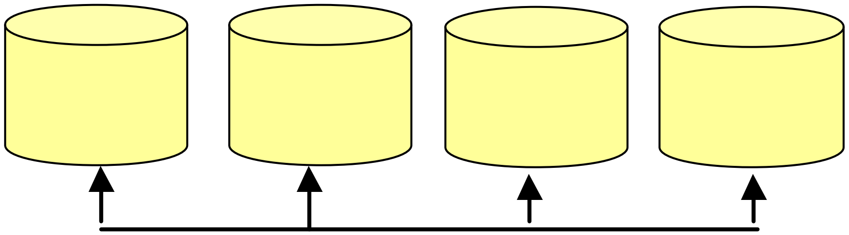
Scenario 2 (Long Distance)

- Servers on different coasts, and data needs to be consistent between them
  - Fast local access
  - Catastrophic failure
  - Data partitioned by region



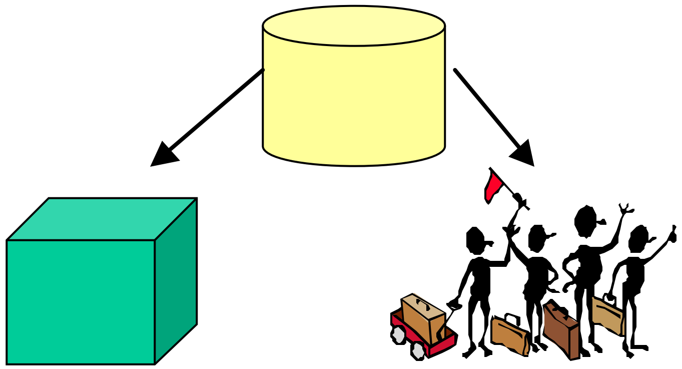
Scenario 3 (Cluster)

- Several servers in a cluster instead of a big mainframe
  - Load balancing
  - Fault resilience
  - Scalability



Scenario 4 (Software Solution)

- Here the database has little or no action in the replication process
  - Data warehouse
  - Mobile users
  - Reporting



PostgreSQL Replication Projects

- **Usogres** (Tesuichi Hosokawa)
- **eRServer** (Vadim Mikheev and Thomas Lockhart @ PostgreSQL, Inc.)
- **PostgreSQL Replicator** (Matteo Cavalleri, Rocco Prudentino @ IRCCS)
- **Postgres-R** (Bettina Kemme, Win Bausch, Gustavo Alonso, Michael Baumer, Ignaz Bachman, and others @ ETH Zurich)

Usogres

- Type: Full, Read Only, Pre-Process
- Location: usogres.good-day.net/
- Description:
  - Real-time backup utility
  - Replicates data via pre-postmaster
  - Supports only one main server and backup server

eRserver

- Type: Partial, Read Only, Transactional
- Location: http://www.erserver.com/
- Description:
  - Supports snapshots (bundles changes)
  - Uses SPI, Perl and PG_Perl interface
  - SyncIDs are used to keep track of the slave data updates

eRServer Desctiption Cont…

- Uses a replication table on master to capture updates via trigger then synchronization can be manual via command line or by number of snapshots
- Only one slave and no fail over support

PostgreSQL Replicator

- Type: Partial, Peer-to-Peer, Async
- Location: pgreplicator.sourceforge.net
- Description:
  - Robust update conflict detection and resolution mechanism
  - Creates a set of Replication Schema Tables (RST) to store the replication rules
  - Uses SP to dynamically capture triggers and auxiliary tables, and act as an interface between the DBA and the the replication engine

PostgreSQL Replicator Cont...

- TCL replication daemon running on each system in the distribution allows database synchronization to be started from any site at any time.
- Uses triggers written in PL/TCL. The TCL daemon uses PostgreSQL connectivity API and TCL/DP libraries for communication over TCP/IP

Postgres-R

- Type: Embedded, Peer-to-Peer, Sync
- Location: gborg.postgresql.org
- Description:
  - Uses a “total order” group communication system (multicasting updates)
  - Shadow copies are used to enforce isolation
  - Propagates tuple changes to decrease processing on remote or query strings if too many tuples changed

Postgres-R Cont…

- Implemented on PostgreSQL 6.4.2
- 4 branches of code (partial and recovery)
- Only replicates one database

Postgres-R Goals

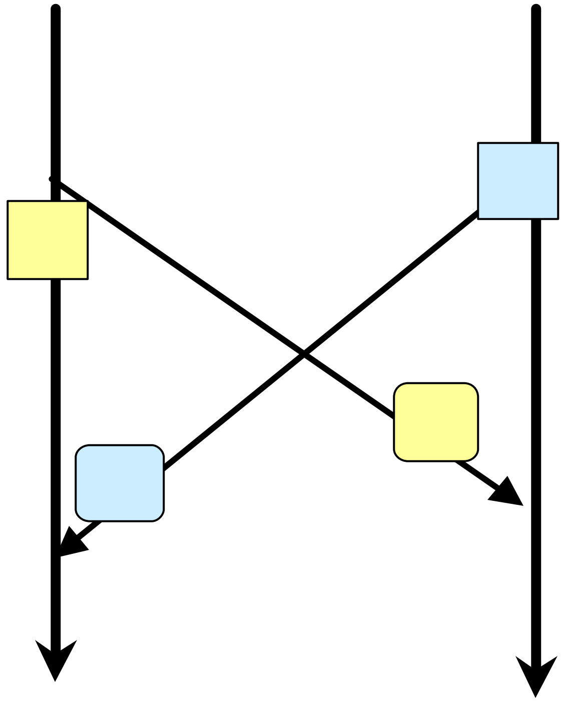
- To develop and apply appropriate techniques in order to avoid previous limitations of synchronous peer-to-peer solutions
  - Good performance (response time + throughput)
  - Consistent and fault tolerant
  - Non-intrusive integration

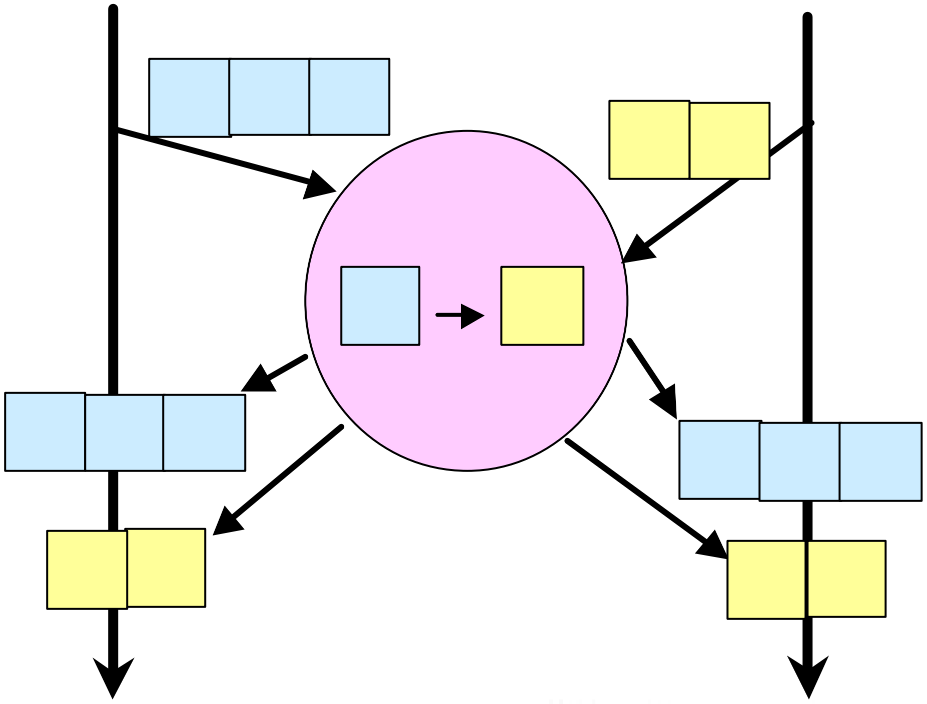
Using Group Communication System

- Multicast
- Delivery order (FIFO, casual, total,etc.)
- Reliable delivery: all nodes vs. all available nodes
- Membership control
- ISIS, Totem, Transis, Horus, Ensemble, Spread

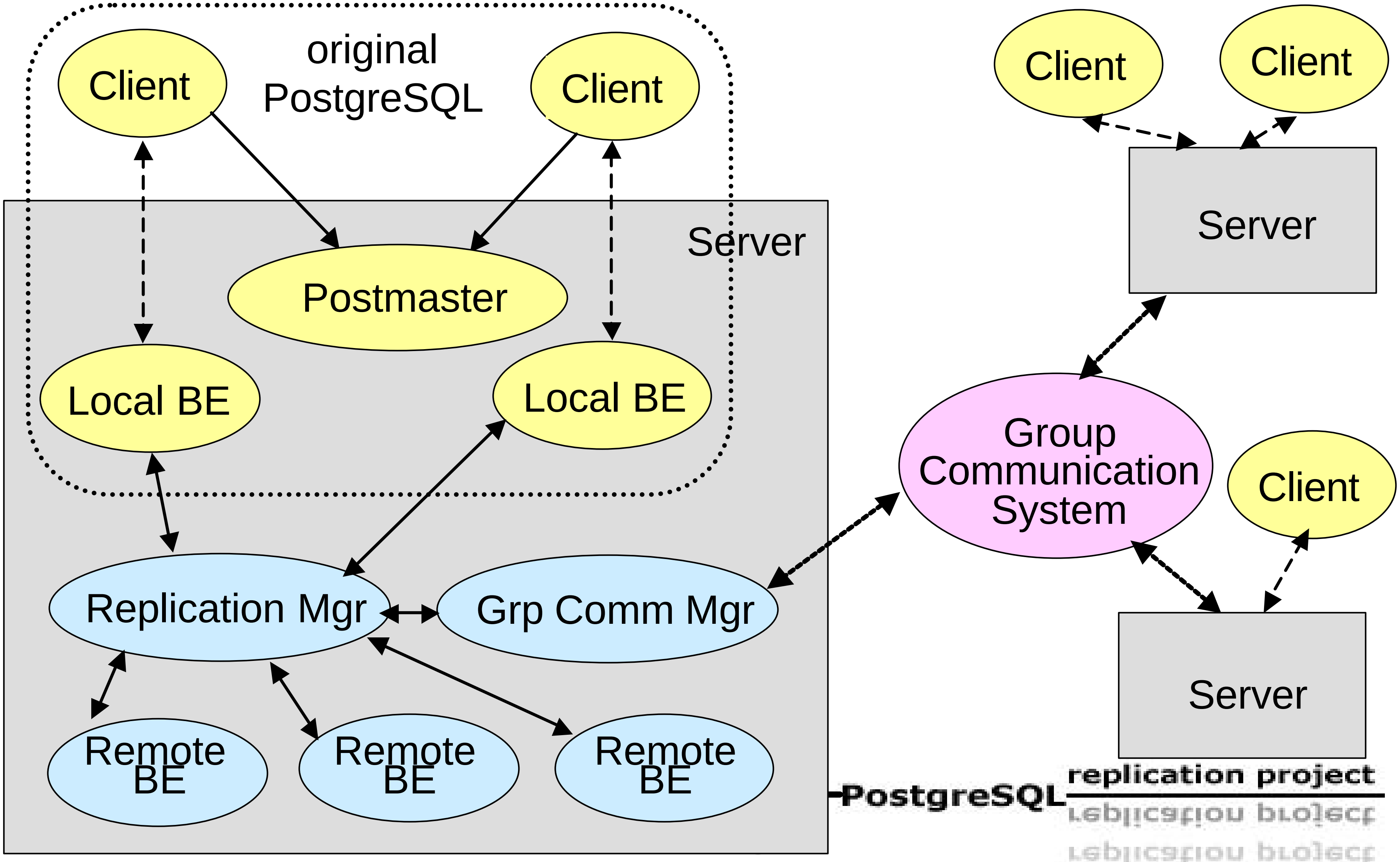
Ordering Transactions

- Two phase commit
- Total order



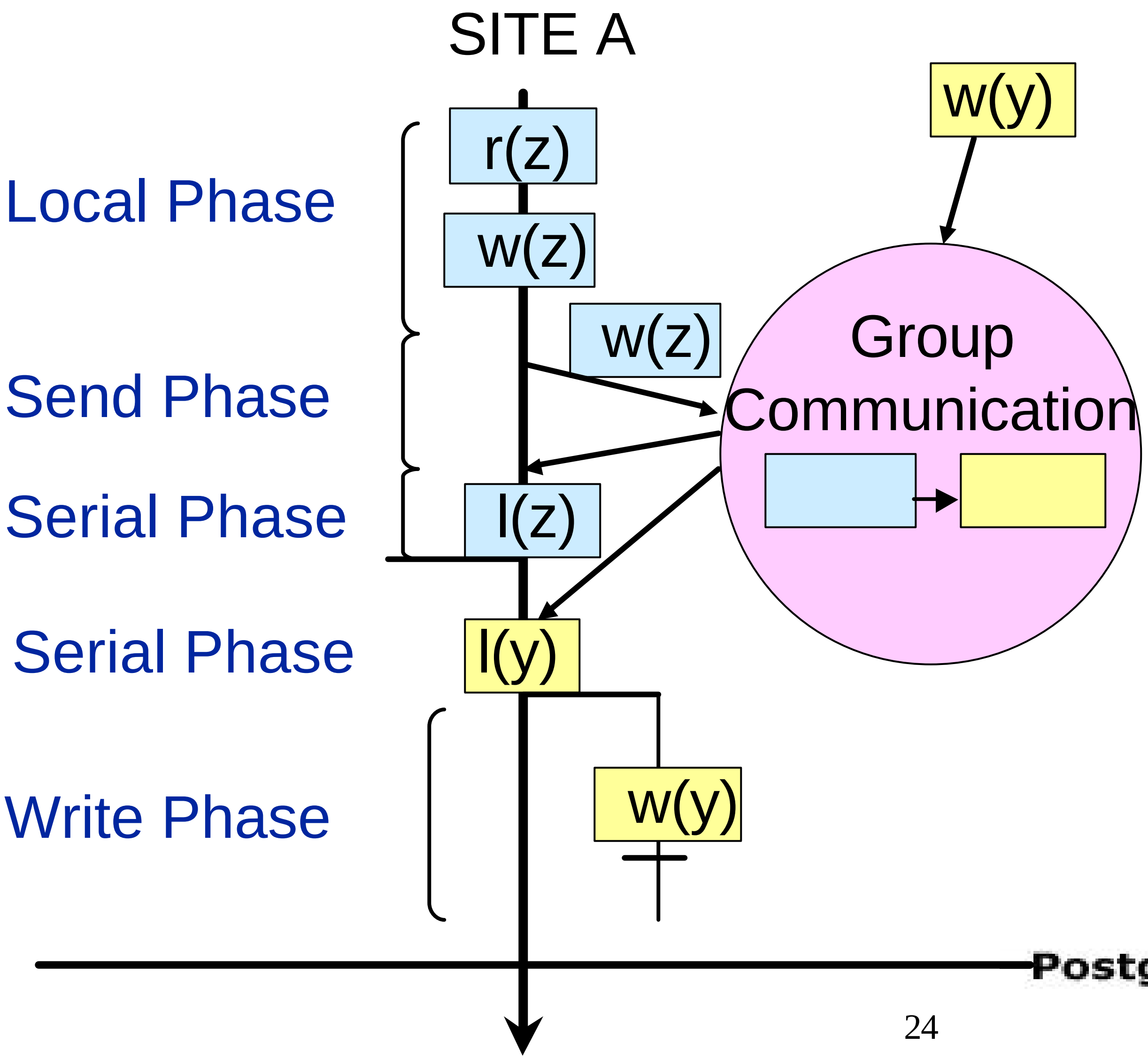


Architecture of Postgres-R



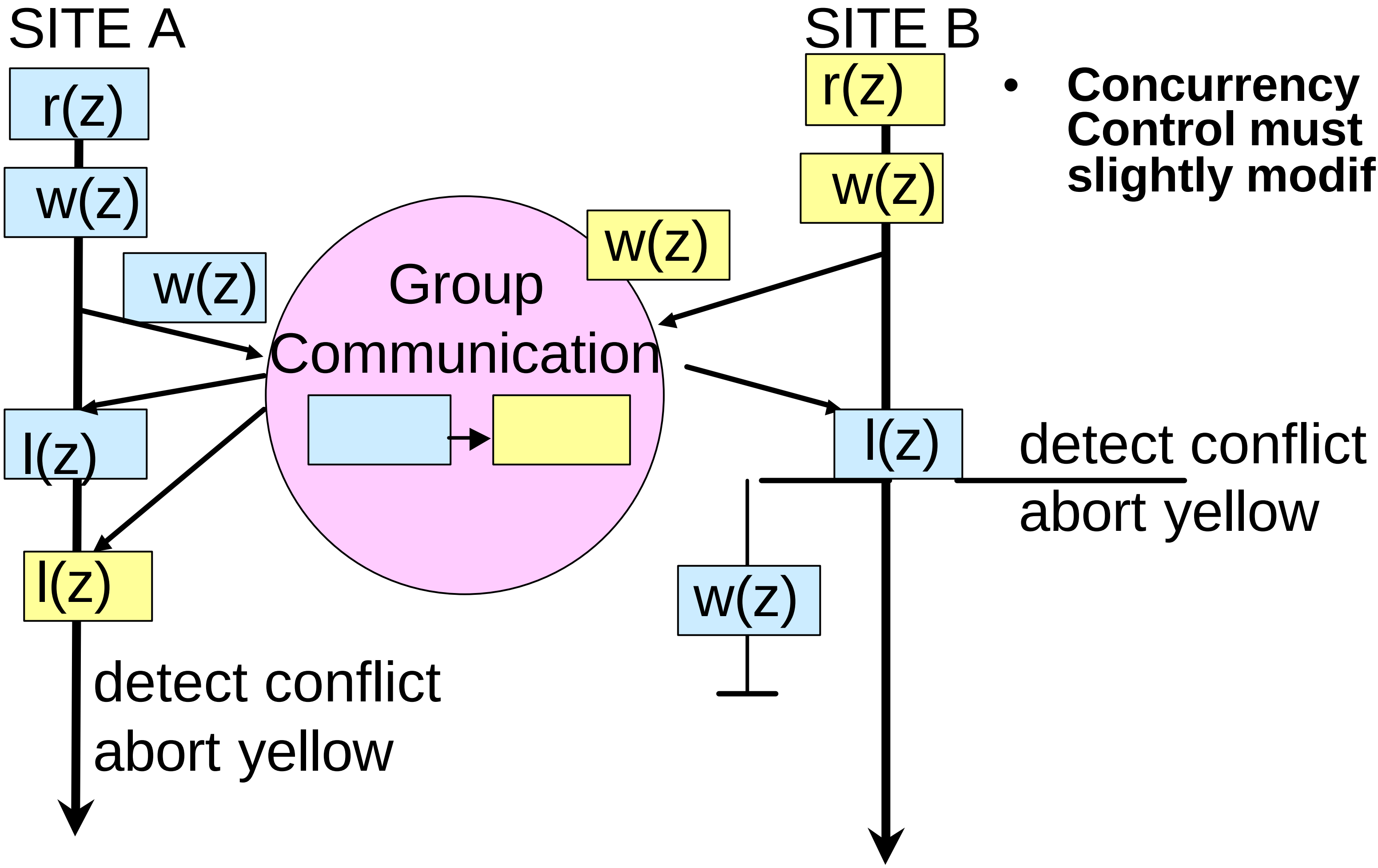
PostgreSQL replication project

Basic Protocol



- All operations are first performed locally at a single site.
- Writes are sent in one message to all sites at the end of transaction.
- Write messages are totally ordered.
- Serialization order obeys total order (same serialization order at all sites).
- When receiving remote writes, they must be applied

Basic Protocol - Conflict



- **Concurrency Control must be slightly modified.**

PGReplication Project

- Goal – to provide a replication solution for PostgreSQL which will meet most needs of users and applications alike.
- Location – gborg.postgresql.org
- Provide information on current research
- Openly discuss all replication projects and share ideas
- Invitation to combine efforts

PGReplication Roadmap

- Take the theories of Postgres-R and implement them into the current version of PostgreSQL using Spread as GCS
- Phase 0 – Full, Read Only, many slaves
- Phase I – Partial, Read Only, Recovery
- Phase II – Peer-to-peer, DDL

Improving PGReplication

- WAL for marshalling
- Using recovery play back mechanism for PITR
- Asynchronous and data partitioning
- Bulk inserts
- Sequences
- Different architectures

Conclusions

- Understanding replication characteristics can help determine the correct solution to a problem.
- Many solutions for replication not one will fit all needs, so we need to be able to support all of them.