PGConf.Russia 2023

PostgresPro

Пять оттенков шардинга

Обзор современных методов
горизонтального масштабирования
реляционных СУБД

Павел Конотопов
руководитель кластерной группы департамента внедрения и технической поддержки

- 20+ лет в ИТ
- Инженер по отказоустойчивости PostgreSQL
- Последние пять лет работаю с PostgreSQL
- Последние два года работаю в Postgres Professional

Email:
p.konotopov@postgrespro.ru

PostgresPro

Что такое шардирование

- Общепринятой дефиниции не существует
- Техника горизонтального масштабирования
- Заключается в разделении таблиц на более мелкие части
- Распределение частей таблиц по разным серверам (шардам)
- Шард содержит часть данных таблиц
- Запрос может быть выполнен параллельно
- Может быть реализовано различными способами

PostgresPro

Требования

- Консистентность
- Отказоустойчивость
- Соответствует какому-то типу нагрузки
  - OLTP
  - OLAP
  - HTAP
- Удобство использования
- Простота администрирования

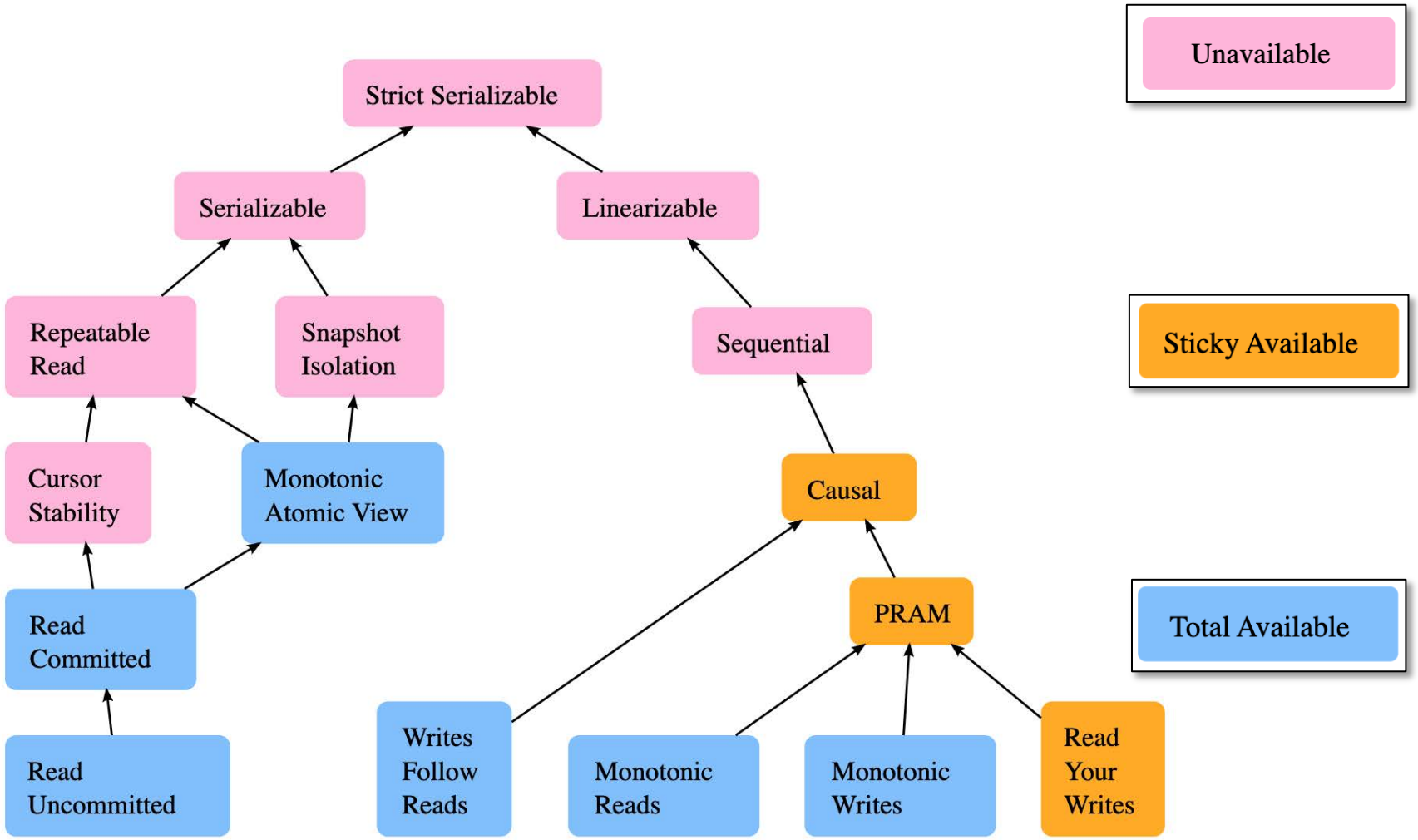
PostgresPro

Терминология

- **Сериализуемость (serializizibility)**
  - Уровень изоляции
  - каждая операция записи или чтения будет возвращать результаты, соответствующие какой-то линейной последовательности операций, которые могут происходить одновременно
  - согласованность данных в контексте *параллельного* выполнения транзакций
- **Линеаризуемость (linearizability)**
  - Уровень консистентности
  - Тоже самое, что сериализуемость
  - Порядок должен учитывать реальное время выполнения
  - Гарантии не для транзакций, а для простых операций над одним объектом
- **Строгая сериализуемость (strict serializable)**
  - Сочетание
    уровня изоляции **serializizibility** и
    уровня консистентности **linearizability**

PostgresPro

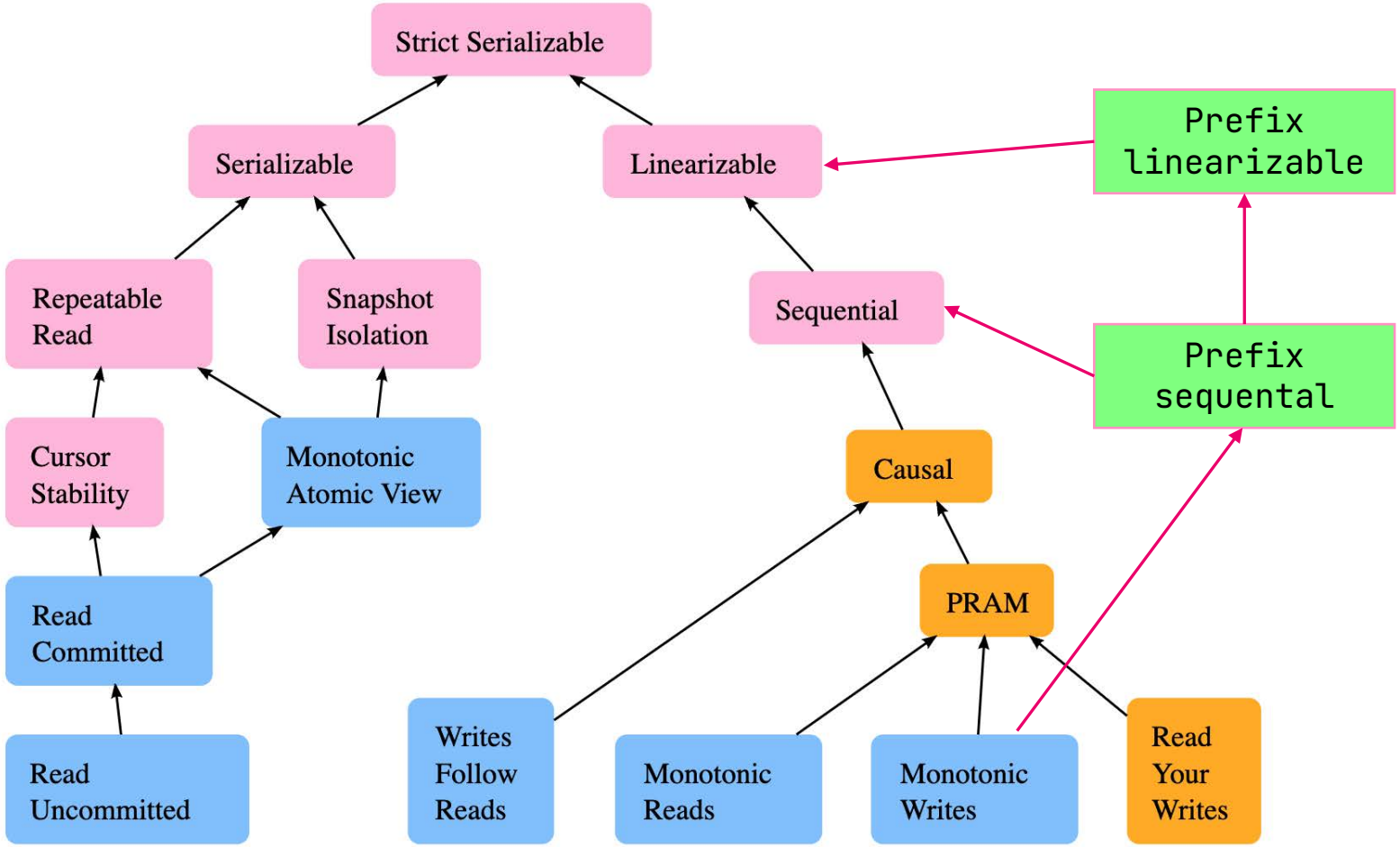
Модели конситентности



- Недоступно при некоторых типах сетевых сбоев. Узлы должны приостановить работу для обеспечения безопасности.
- Доступен на всех узлах, не имеющих сбоев, пока клиенты общаются только с теми же серверами, а не переключаются на новые.
- Доступен на каждом неисправном узле, даже когда сеть полностью отключена.

PostgresPro

Модели конситентности



- **Одиночный сервер**
  - serializable
  - level linearizable
- **С синхронной репликой** (remote_apply)
  - serializable
  - write linearizable
  - read sequential
- **С асинхронной репликой**
  - Serializable
  - Read-write linearizable
    Read-only prefix linearizable

PostgresPro

Использование и администрирование

- Чем меньше ограничений тем лучше
- Чем меньше дополнительной инфраструктуры тем лучше
- Доступ в кластер через любой узел
- Минимум специального SQL диалекта
- Максимум совместимости с одиночным сервером PostgreSQL
- Не нужно думать о шардинге с точки зрения SQL

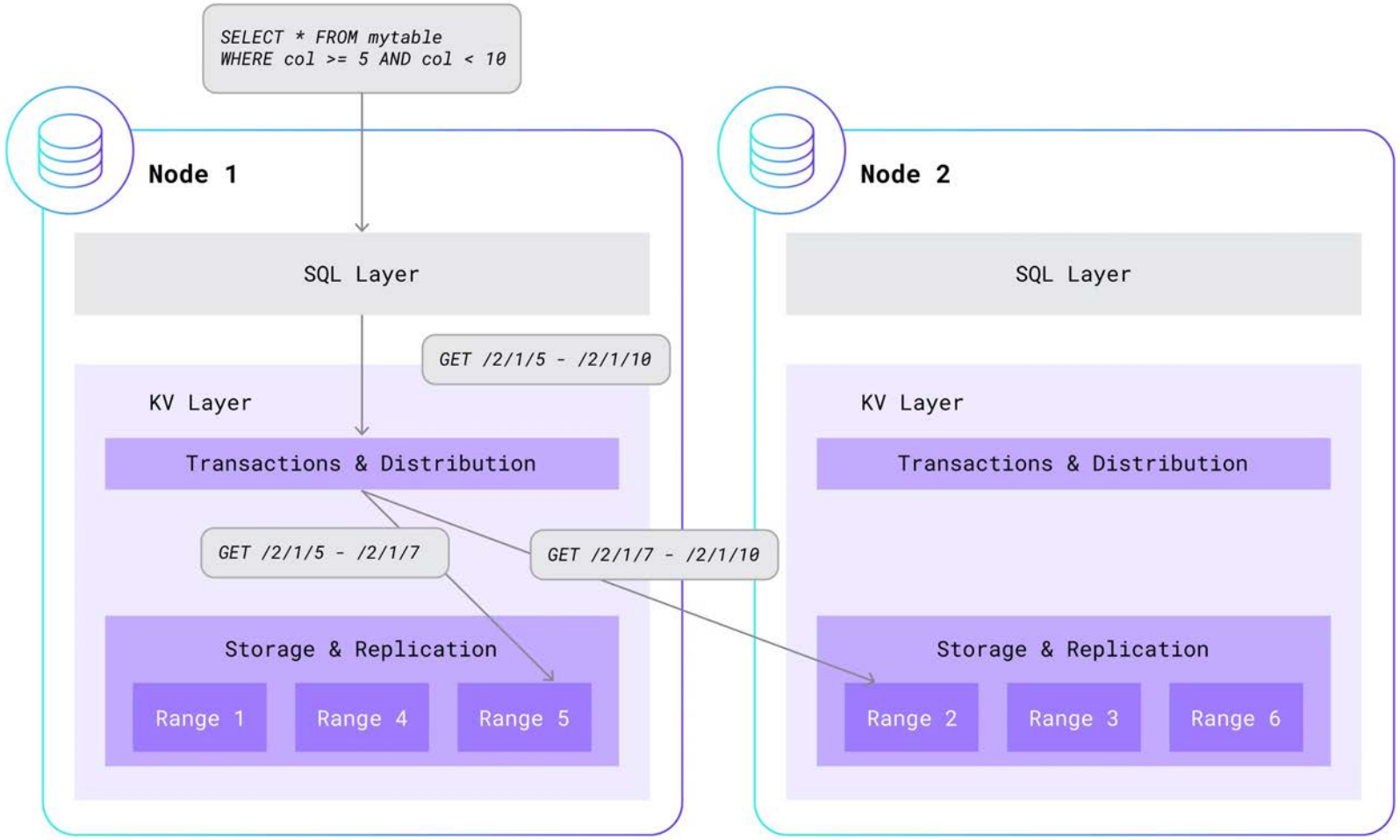
- Наличие cli-утилиты
- Мониторинг
- Удобство развертывания
  - on-premise, cloud, containers

CockroachDB

- Предназначен для геораспределенных данных
- Встроенная отказоустойчивость
  - Один бинарный файл
- Диалект SQL
  - DistSQL
  - Частично совместим с PostgreSQL
- Не основан на коде PostgreSQL
- Хранилище данных - поверх K/V RocksDB (своя реализация - Pebble)
- Предназначена OLTP, но не слишком быстро

- Модель консистентности:
  - Почти strict serializable
    - stale reads невозможен
    - causal reverse возможен

CockroachDB



PostgresPro

CockroachDB, ограничения

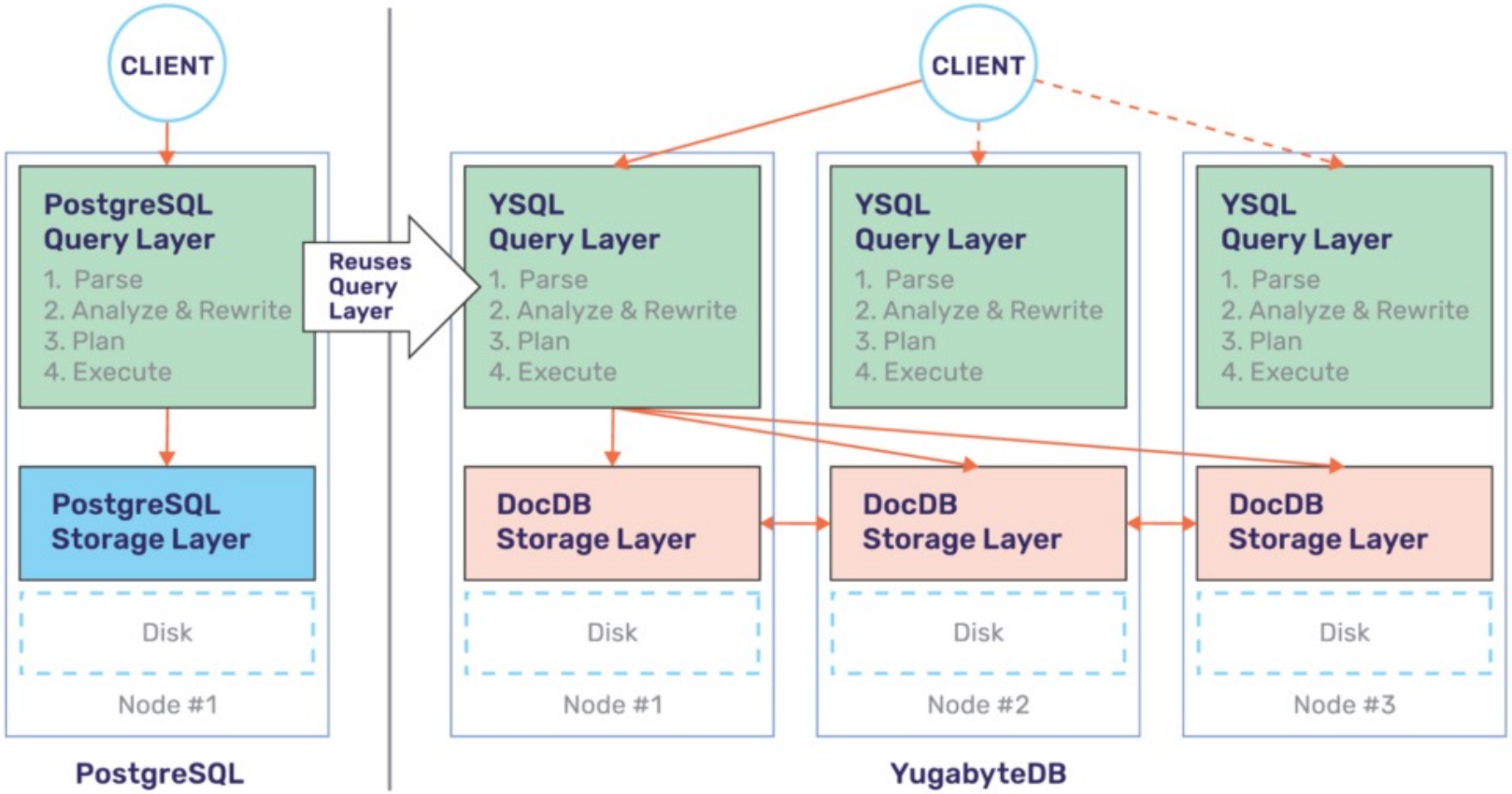
- Хранимые процедуры
  - Поддерживаются user-defined functions
- Триггеры
- События (listen/notify)
- Полнотекстовый поиск и функции
- Удаление первичного ключа
- XML функции
- Ограничение прав доступа для колонок
- Создание БД из шаблона
- Удаление одной партиции
- FDW

YugabyteDB

- Очень похожа на CockroachDB по архитектуре
- Предназначена для геораспределенных данных
- Встроенная отказоустойчивость
  - отказ от доступности в пользу согласованности
- Диалекты SQL
  - YSQL (совместим с PostgreSQL)
  - YCQL (полуреляционный язык, основан на Cassandra Query Language)
- Частично построена на коде PostgreSQL 11
  - Parser, Analyser, Planner, Executor
- Хранилище данных - поверх K/V DocDb (RocksDB)
- Предназначена для OLTP, но не слишком быстро
- отказ от доступности в пользу согласованности

- YSQL поддерживает уровни изоляции
  - Serializable,Snapshot и Read Committed
- YCQL поддерживает только изоляцию Snapshot
- Модель консистентности
  - Почти strict serializable

YugabyteDB



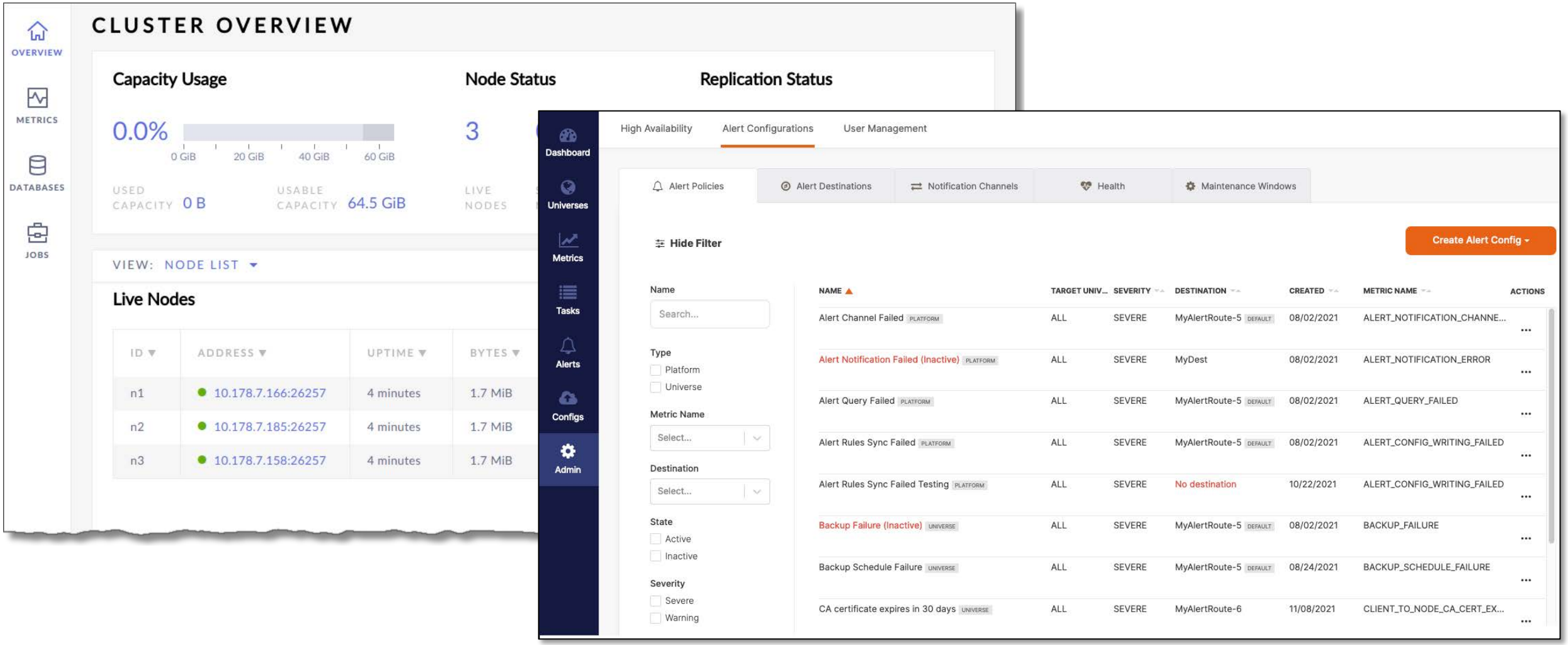
How YugabyteDB reuses PostgreSQL query layer

PostgresPro

YugabyteDB, ограничения

- Кодовая база PostgreSQL 11
- Блокировки уровня таблицы
- Наследование таблиц
- Констрейнты (exclusion, deferrable)
- GiST индексы
- GIN индексы по нескольким колонкам
- События (Listen/Notify)
- XML функции
- Ограничения ключей Primary/Foreign на foreign таблицах
- GENERATED ALWAYS AS STORED
- CREATE ACCESS METHOD
- CREATE SCHEMA with elements
- DDL в транзакции

Администрирование

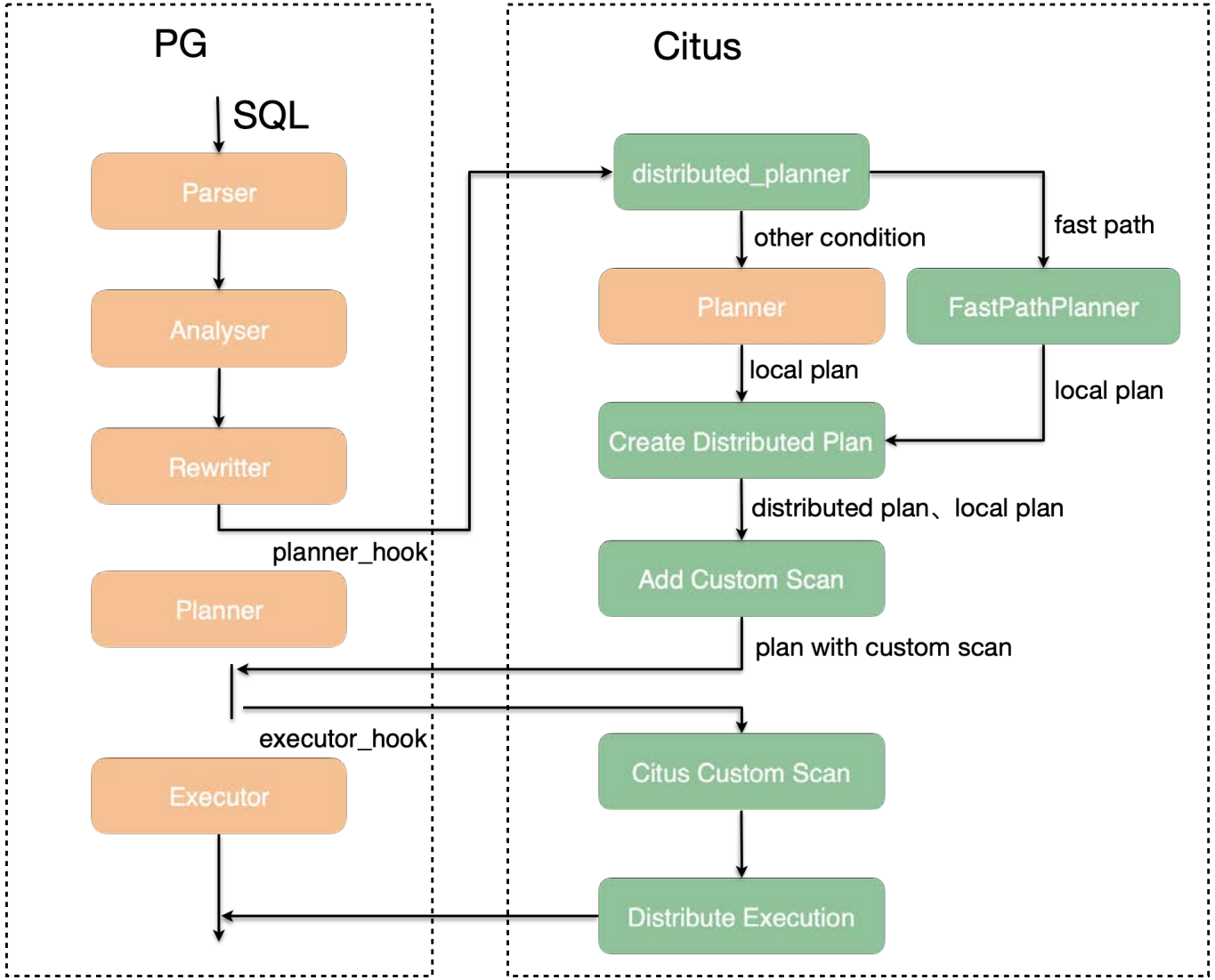


PostgresPro

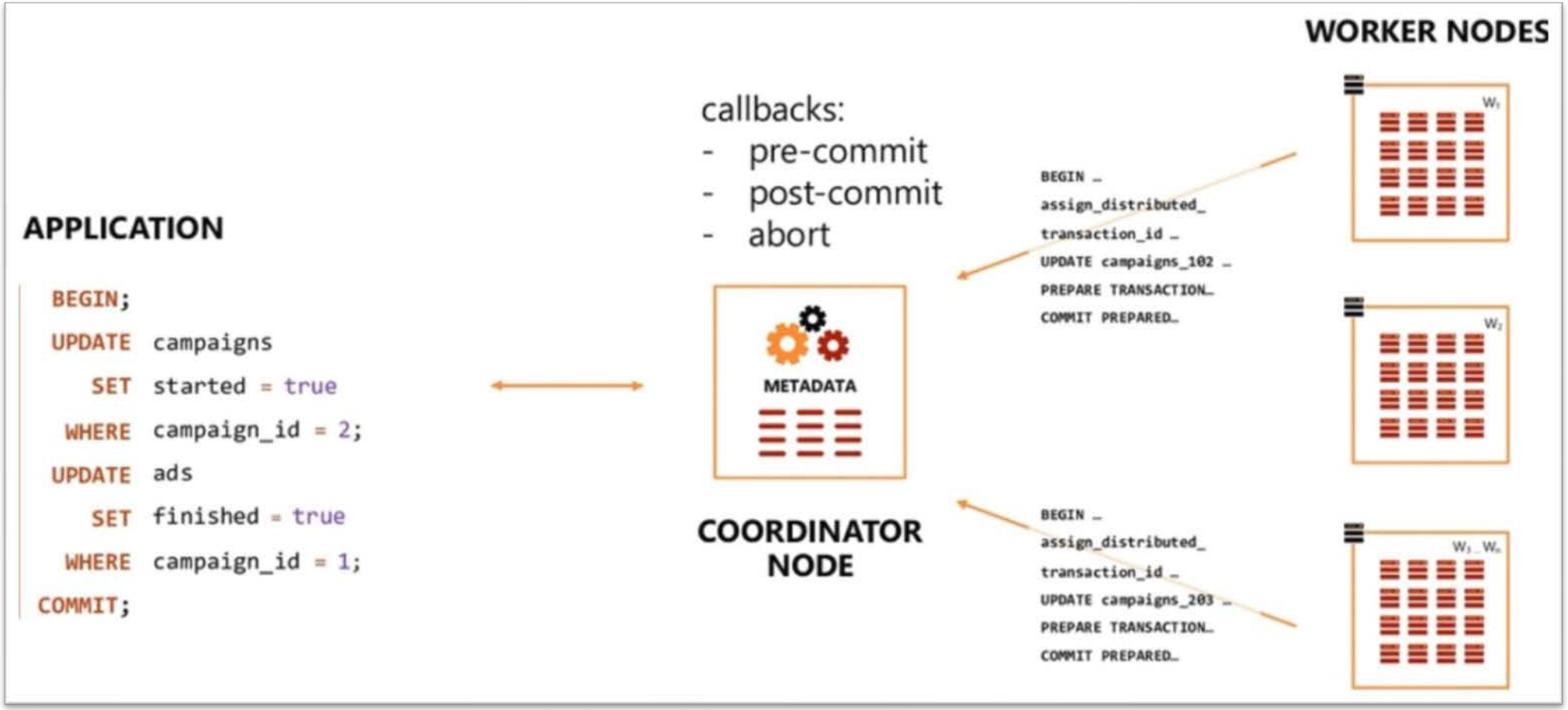
CitusDB

- Предназначен для локализованных данных
- Встроенной отказоустойчивости нет
  - Узел координатор
  - Узлы хранения данных
  - Поддержка в Patroni 3.x.x, k8s
    - Требует дополнительной инфраструктуры
  - Избыточность данных
- PostgreSQL внутри PostgreSQL
  - Начинался до ванильного партицирования
  - Целились в OLAP
- Расширение PostgreSQL
  - Широко использует хуки PostgreSQL
- Предназначен для OLAP
  - Развивается в сторону OLTP
- Модель согласованности
  - Snapshot в рамках одного узла + 2PC
  - Read uncommited – уровень изоляции
  - Linearizable – уровень косистентности

CitusDB


PG
SQL
Parser
Analyser
Rewritter
planner_hook
Planner
executor_hook
Executor
Citus
distributed_planner
other condition
fast path
Planner
FastPathPlanner
local plan
local plan
Create Distributed Plan
distributed plan、local plan
Add Custom Scan
plan with custom scan
Citus Custom Scan
Distribute Execution

CitusDB


WORKER NODES
callbacks:
- pre-commit
- post-commit
- abort
APPLICATION
BEGIN;
UPDATE campaigns
SET started = true
WHERE campaign_id = 2;
UPDATE ads
SET finished = true
WHERE campaign_id = 1;
COMMIT;
METADATA
COORDINATOR NODE
BEGIN …
assign_distributed_
transaction_id …
UPDATE campaigns_102 …
PREPARE TRANSACTION…
COMMIT PREPARED…
BEGIN …
assign_distributed_
transaction_id …
UPDATE campaigns_203 …
PREPARE TRANSACTION…
COMMIT PREPARED…

PostgresPro

CitusDB, достоинства и недостатки

- Только SQL интерфейс для управления кластером
- Слабая модель согласованности
  - ACID на уровне отдельных узлов
  - Нет «честных» распределенных транзакций
- DDL только через координатор
- Нет консистентных резервных копий

- Ориентирован на OLAP, движется в сторону OLTP
  - Fastpath – быстрый разбор запроса и отправка его на нужный узел
  - Shuffle join через libpq
  - DML через любой узел (с 11 версии)
  - Нет мультиплексирования (интерконнект)
    - Требуется пулер соединений между узлами кластера

PostgresPro

Greenplum

- Распределенная массивно-параллельная СУБД (MPP)
- Частичная встроенная отказоустойчивость
  - Избыточность данных
- Аналитическая система
- Форк, базируется на PostgreSQL 9.6-12
- Горизонтальное масштабирование
- Интерконнект
- Отказоустойчивость, обеспечивается созданием зеркал каждого логического сегмента и резервного мастер-сервера.
- Запросы – через координатор
- Возможность выполнять локальные и распределённые JOIN.

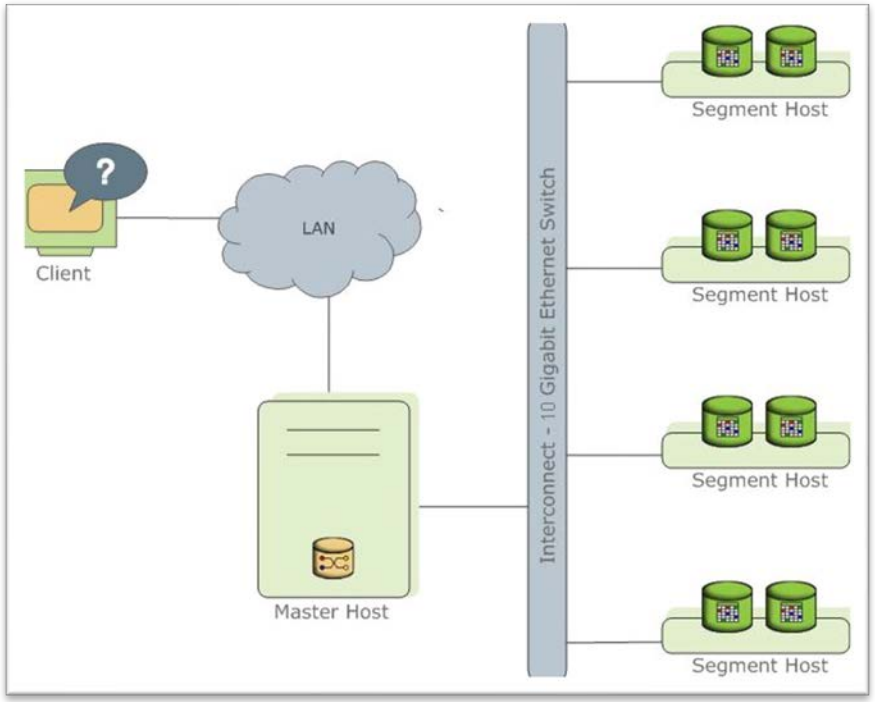
- Два уровня изоляции транзакций
  - Read commited и repeatable read

- Модель согласованности
  - Snapshot isolation
  - Linearizable

PostgresPro

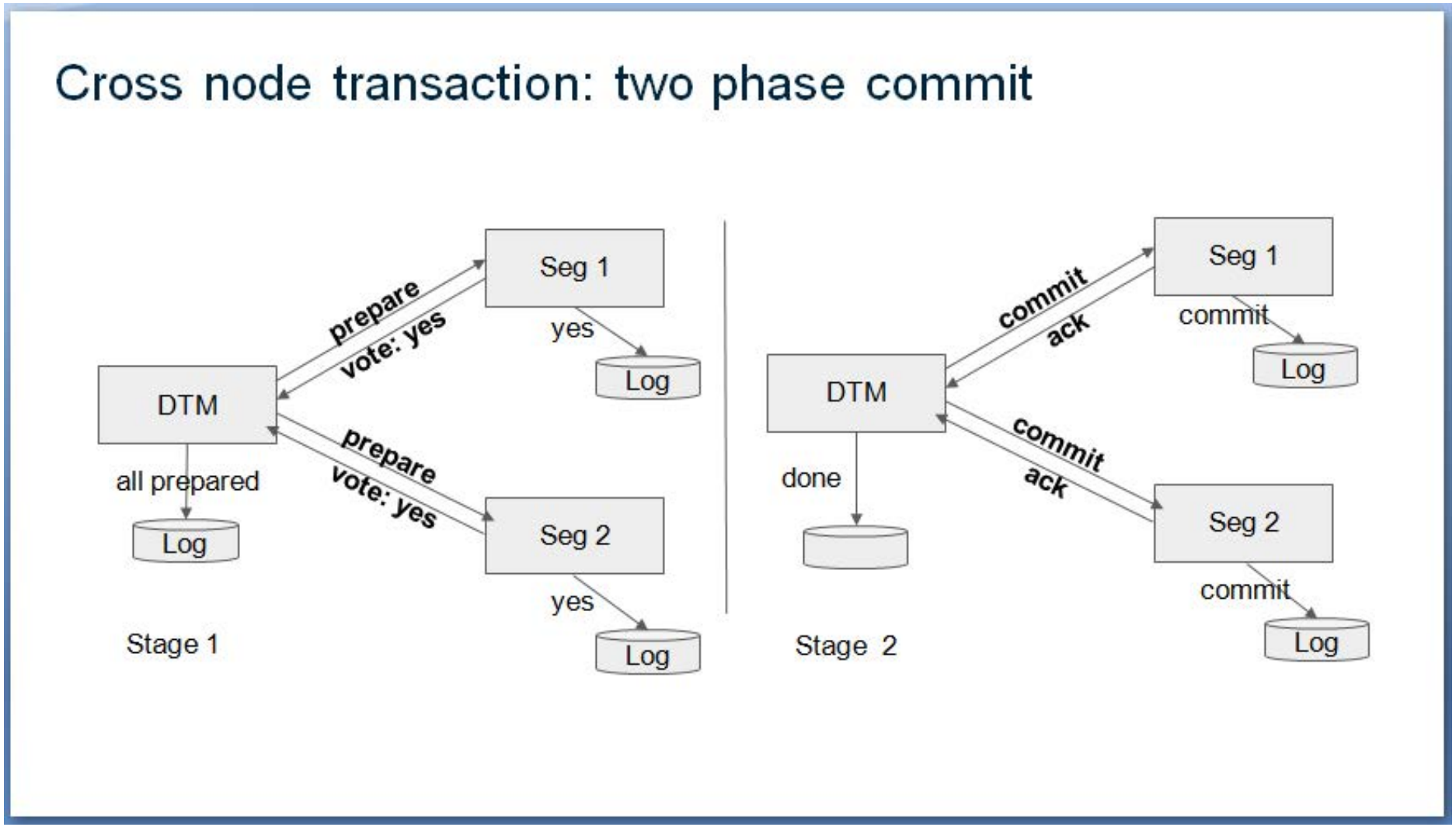
Greenplum

- **Мастер-сервер**
  - точка входа в Greenplum
  - координирует свою работу с сегментами
  - распределяет нагрузку между сегментами
  - не содержит пользовательских данных
- **Сервер-сегмент**
  - Хранит и обрабатывает данные
  - Отдает результаты мастеру
- **Интерконнект (внутрикластерный протокол)**
  - сетевое соединение для связи между отдельными экземплярами PostgreSQL



PostgresPro

Greenplum



PostgresPro

Greenplum, достоинства и недостатки

- Координатор строит распределенные планы выполнения запроса
- Распределенные транзакции через менеджер транзакций (DTM)
- Потеря координатора

- Хорошо поддержаны разные типы JOIN
  - Загрузка узлов кластера вычислительной работой
- cli для управления кластером, миграции данных, мониторинга
- Параллельный бекап-восстановление кластера
  - Access share lock – запрет изменений
  - Инкрементальный бекап
- Встроенная отказоустойчивость сегментов

PostgresPro

SPQR/PGCAT

- Middleware
- Решают проблему для простых OLTP запросов
- Решает проблему роутинга и балансировки
- Пулер соединений
- Мониторинг реплик

PostgresPro

Ванильный PostgreSQL

- Ванильное партицирование + postgres_fdw
- Ручное управление
  - Создать партицированную таблицу на всех узлах
  - Настроить foreign servers
  - Настроить чередование секций на каждом сервере
  - Связать секции через postgres_fdw
- Будет работать медленно
  - плохо работает push-down запросов
  - используется неоптимальный протокол libpq для интерконнекта
  - нет мультиплексирования, большое количество подключений
- Нет 2PC через fdw (идет работа)
- Модель консистентности
  - Snapshot в рамках одного узла
  - Read uncommited
  - Возможно linearizable

PostgresPro

Shardman

- Прозрачное горизонтальное масштабирование
- OLTP нагрузка
- Доступ в кластер через любой узел
- Избыточность данных (отказоустойчивость)
- Целостность данных (распределенные транзакции)

PostgresPro

Shardman

- Расширение
- Секционирование
- postgres_fdw
- Мультиплексирование TCP соединений
- Физическая и логическая репликация
- CSN – commit sequence number (Clock-SI)

PostgresPro

Shardman

- Расширение
- Интегрированы возможности Postgres Pro Enterprise
  - CFS, PTRACK, pgpro_stats, AQO - адаптивный оптимизатор запросов
- Push-down запросов
  - Ассиметичный JOIN (партиция+глобальная таблица)
  - Агрегатные функции
  - Partition wise join (партиция + партиция)
- Доступ через любой из узлов кластера
  - Мультиплексирование соединений
- Консистентный бекап-восстановление
  - Инкрементальное, дельта и PTRACK
- Любая топология кластера
- Единая точка управления - cli утилита
- Модель консистентности
  - Изоляция Repeatable read (snapshot isolation)
  - Консистентность
    - write-only linearizable
    - read-write и read-only - prefix sequential

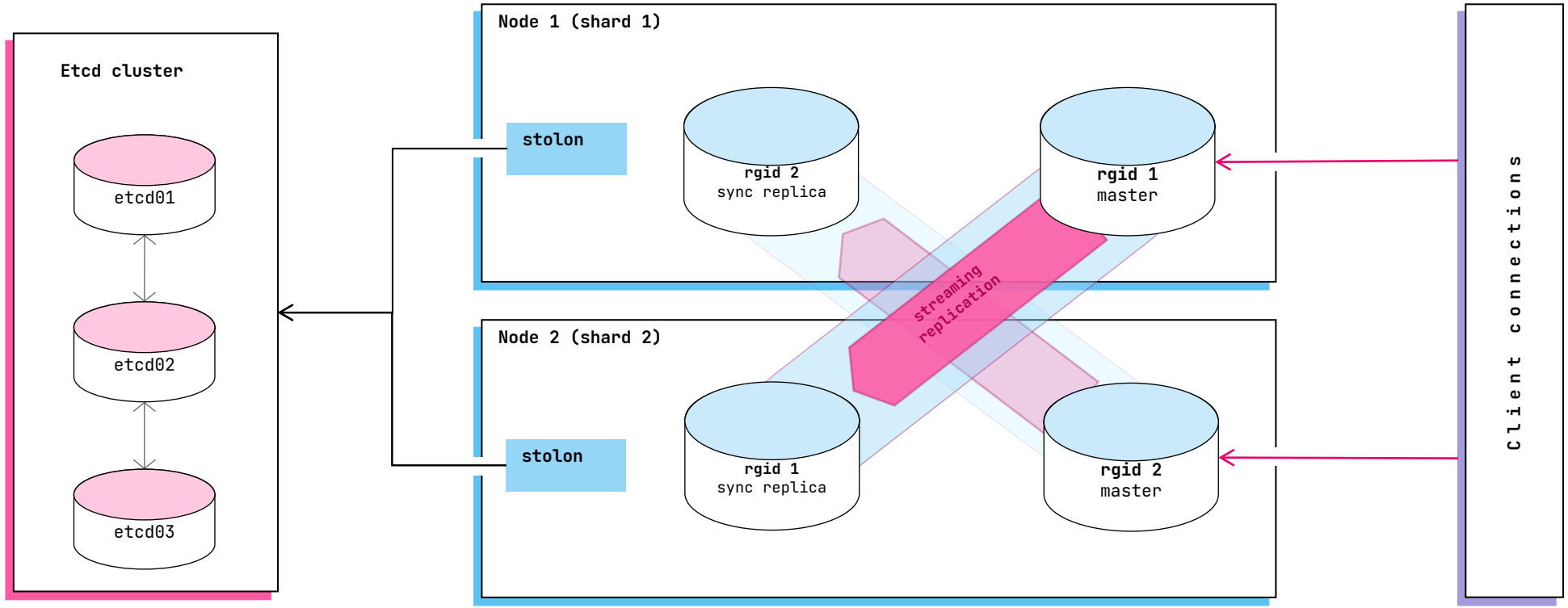
PostgresPro

Shardman

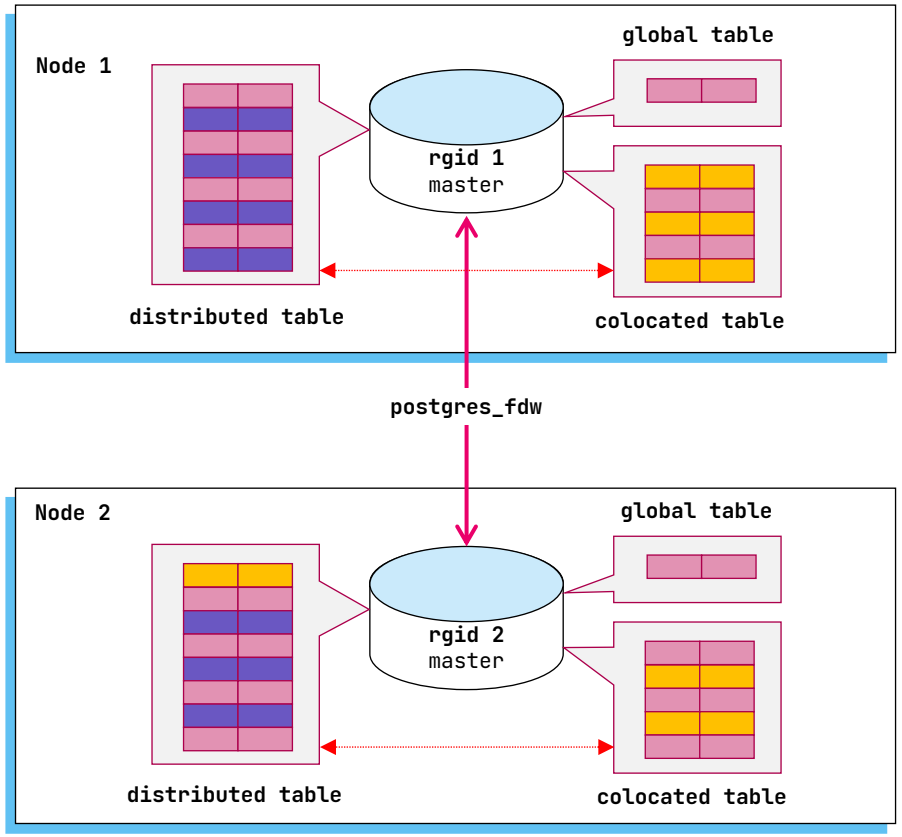
- PostgreSQL 14
- Stolon
- Расширения Shardman и postgres_fdw
- Etcd

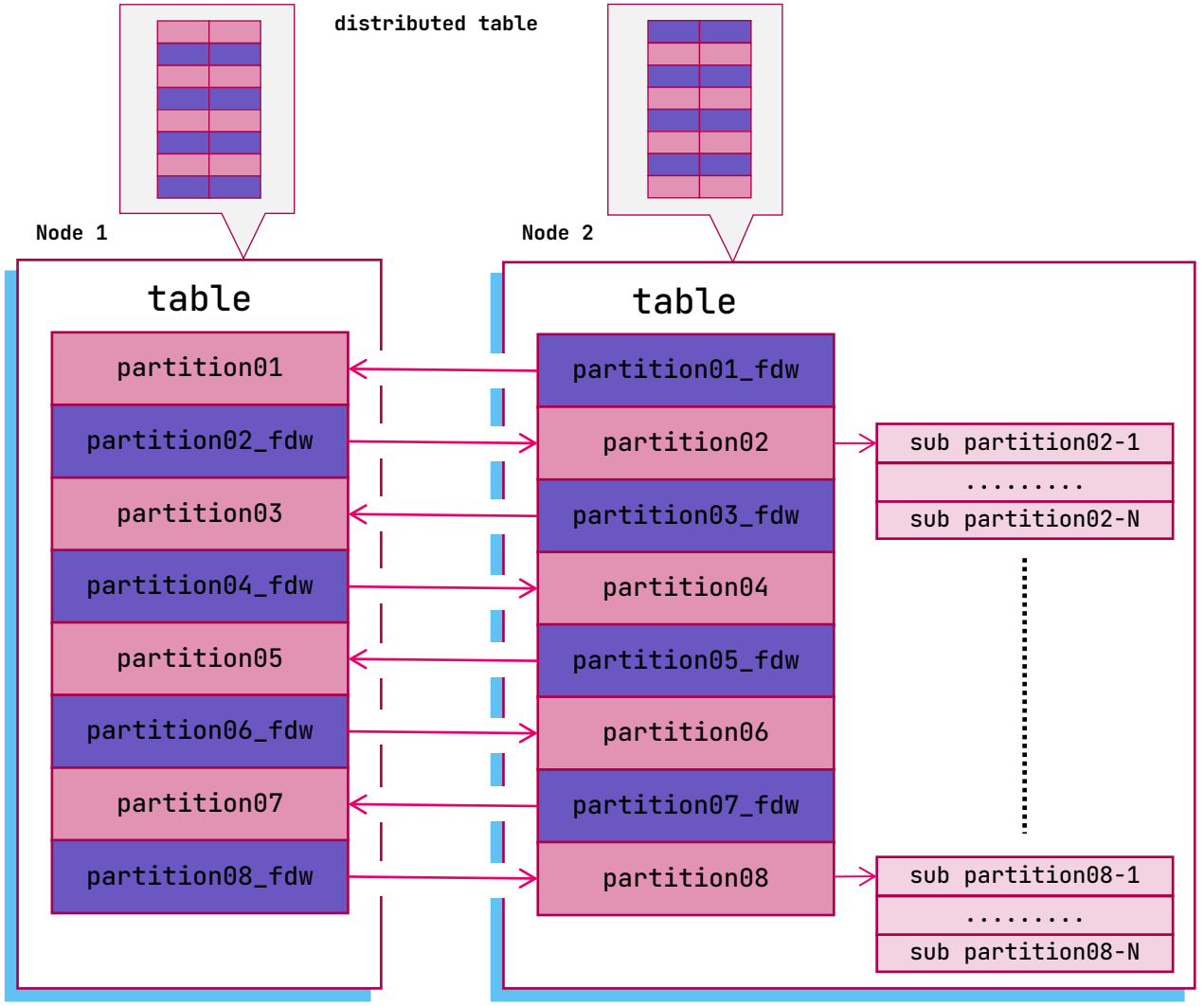
- shardmand
  сервис, отвечающий за кластерное ПО
- shardmanctl
  утилита управления кластером

Shardman

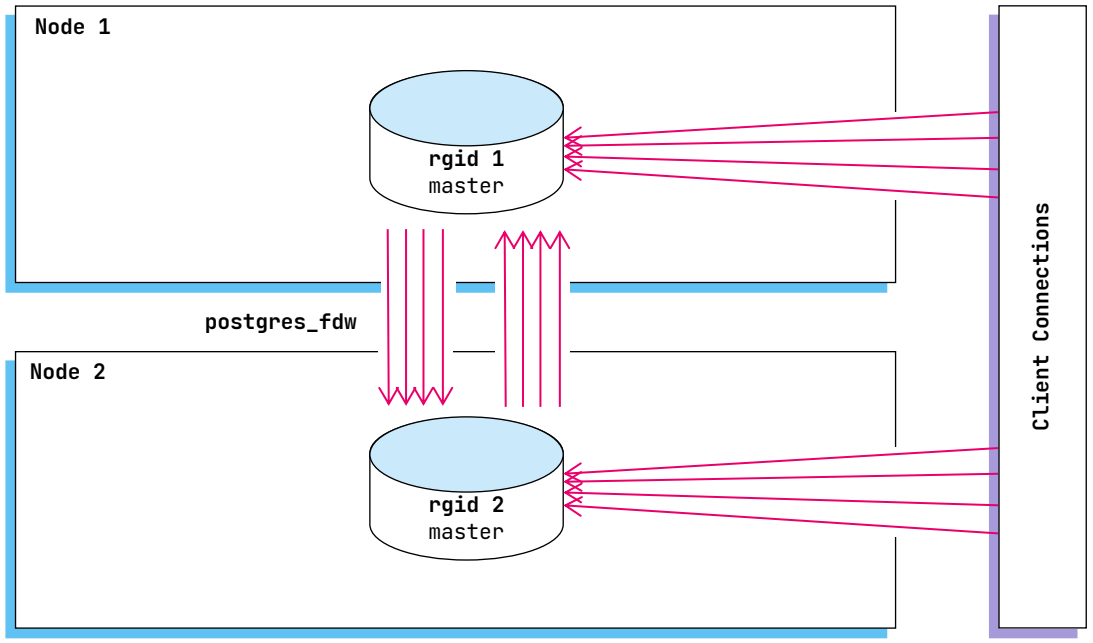

Etcd cluster
etcd01
etcd02
etcd03
Node 1 (shard 1)
stolon
rgid 2
sync replica
rgid 1
master
streaming
replication
Node 2 (shard 2)
stolon
rgid 1
sync replica
rgid 2
master
Client connections

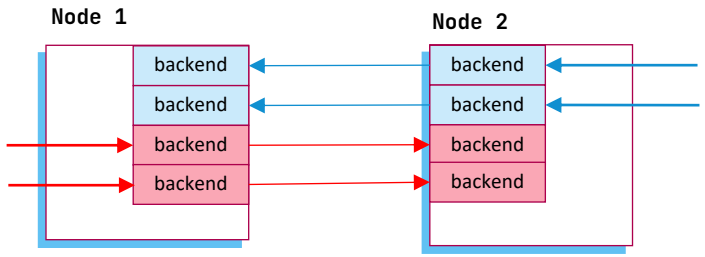
Shardman


Node 1
global table
rgid 1
master
distributed table
colocated table
postgres_fdw
Node 2
global table
rgid 2
master
distributed table
colocated table


distributed table
Node 1
Node 2
table
table
partition01
partition02_fdw
partition03
partition04_fdw
partition05
partition06_fdw
partition07
partition08_fdw
partition01_fdw
partition02
partition03_fdw
partition04
partition05_fdw
partition06
partition07_fdw
partition08
sub partition02-1
.....
sub partition02-N
sub partition08-1
.....
sub partition08-N

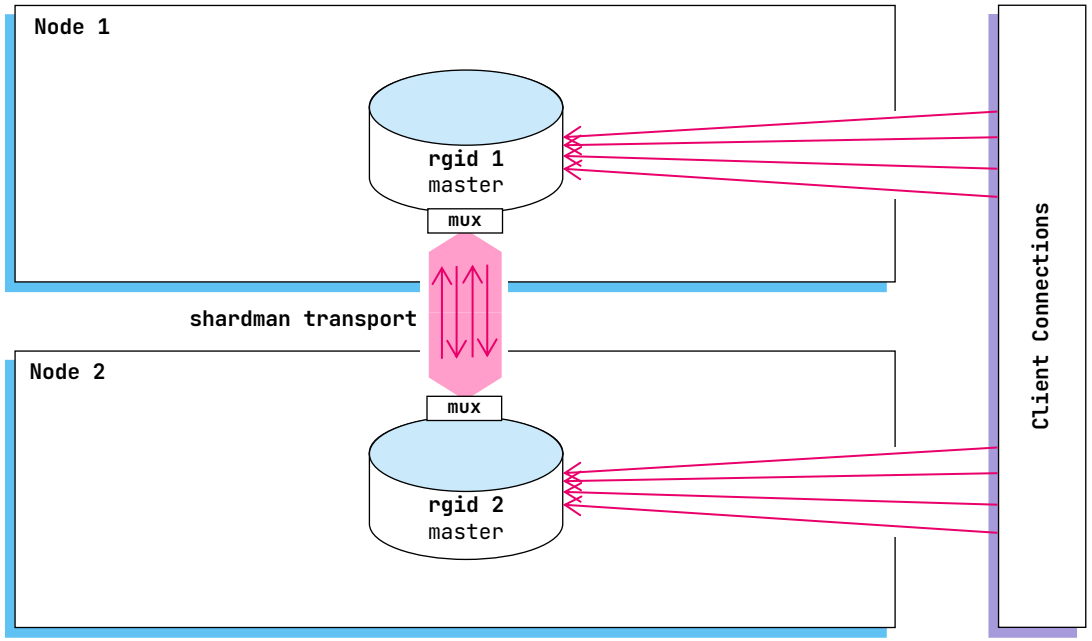
Shardman

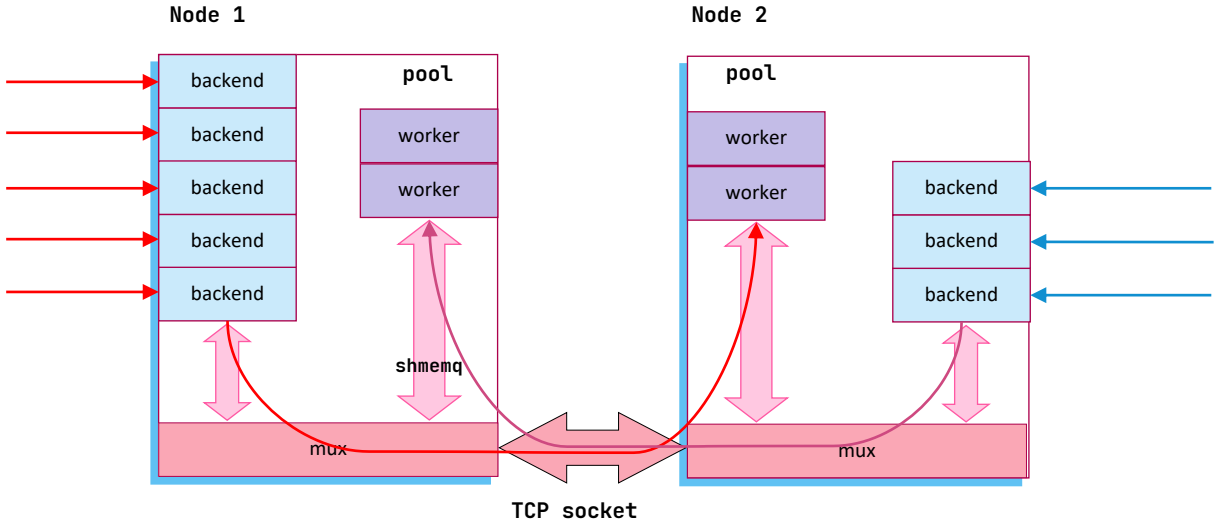




Общее число подключений в кластере – M x N
M – количество подключений,
N – количество узлов

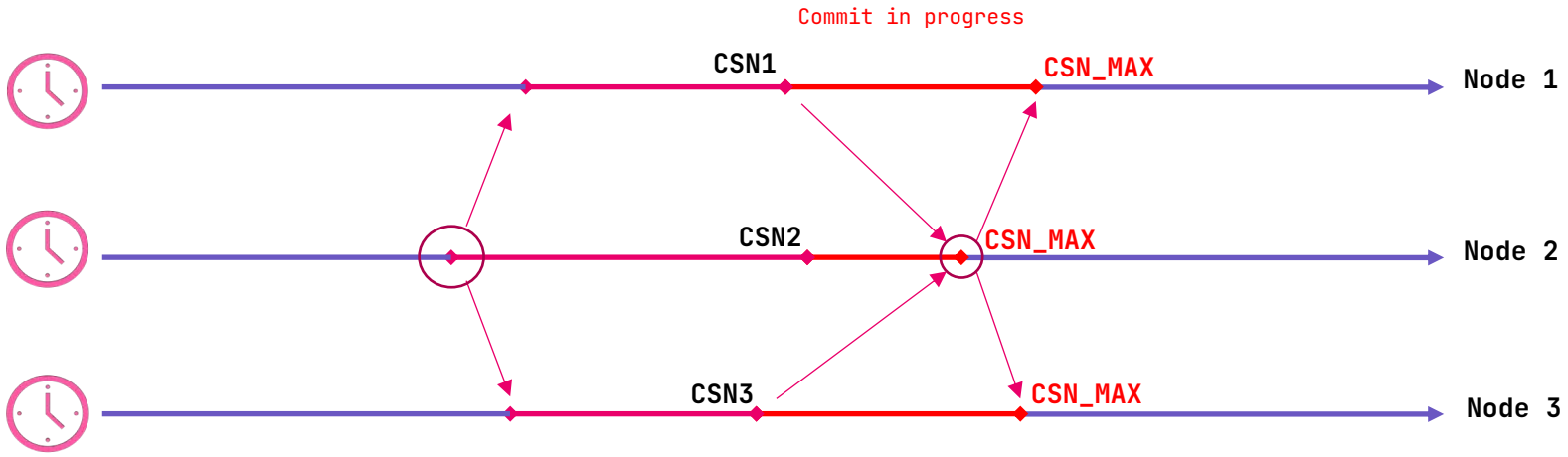
Shardman






```
Общее число подключений кластере – M+(N x N)
Общее число процессов в кластере – M+(N x W)
        M – количество подключений,
        N – количество узлов,
        W – количество воркеров
```

Shardman



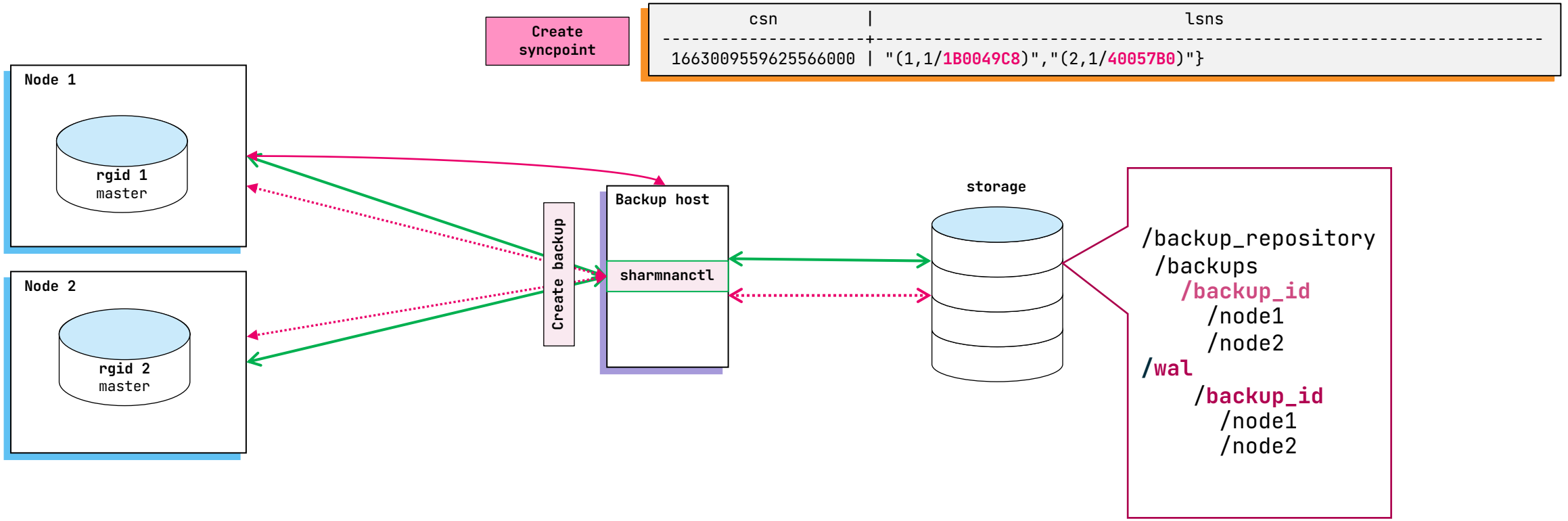
CSN_MAX = MAX(CSN1,CSN2,CSN3)

CSN - время
CSN(LSN_NODE1,LSN_NODE2,LSN_NODE3)

```
# CALL shardman.create_syncpoint(NULL, NULL);

         csn          |                                    lsns
----------------------+-------------------------------------------------------------
 16630095596255660000 | {"(3,0/FD005D60)","(1,1/1B0049C8)","(2,1/40057B0)"}
```

Shardman


Create syncpoint
csn | lsns
1663009559625566000 | "(1,1/1B0049C8)","(2,1/40057B0)"}
Node 1
rgid 1
master
Node 2
rgid 2
master
Create backup
Backup host
sharmnanctl
storage
/backup_repository
/backups
/backup_id
/node1
/node2
/wal
/backup_id
/node1
/node2

PostgresPro

Shardman

- shardmanctl – cli утилита управления cluster/config/init/shards/nodes/rebalance/backup/load/upgrade/update/status
- Инициализация
- Добавление и удаление узлов
- Ребалансировка
- Бекап и восстановление
- Обновление до новой версии
- и т.д.

PostgresPro

Shardman, ограничения

- Распределение данных – только хэшу
- Все формы ALTER TABLE невозможны, кроме:
  - Изменение владельца таблицы,
  - SET/DROP NOT NULL,
  - ADD/DROP COLUMN, кроме type serial (автоинкремент)
  - Добавление и удаления индексов
- Нельзя создавать распределенные временные таблицы
- Нельзя изменить количество секций распределенных таблиц
- Невозможно переименовывать глобальные роли
- Только один тип хранения данных – таблицы PostgreSQL
- Не позволяет использовать реплики для чтения данных
- Транспорт работает только для SELECT

PostgresPro

Таблица сравнения

	CockroachDB	YugabyteDB	Greenplum	CitusDB	Shardman
Уровень изоляции	Почти strict serializable	Почти strict serializable	snapshot isolation	read uncommited	snapshot isolation
Уровень консистентности	linearizable	linearizable prefix sequental	linearizable	linearizable	linearizable prefix sequential
Доступ в кластер	Любой узел	Любой узел	Координатор	Координатор (DDL), Любой узел (DML)	Любой узел
Отказоустойчивость	Да/Auto HA	Да/Auto HA	На уровне датанод/Auto HA/Mirroring	Нет/Patroni/Auto HA	Да
Тип нагрузки	OLTP	OLTP	OLAP	OLAP->HTAP	OLTP->HTAP
Удобство администрирования	UI/cli	UI/cli	cli	SQL интерфейс	cli
Наличие дополнительной инфраструктуры	Не требуется	Не требуется	Не требуется	Нет/DCS	DCS

PostgresPro

Ссылки

- **Consistency models**
  - https://jepsen.io/consistency
  - https://www.cockroachlabs.com/blog/consistency-model
- **CockroachDB**
  - https://www.cockroachlabs.com/docs
- **YugabyteDB**
  - https://docs.yugabyte.com
- **CitusDB**
  - https://docs.citusdata.com
- **SPQR**
  - https://github.com/pg-sharding/spqr
- **Pgcat**
  - https://github.com/postgresml/pgcat
- **Shardman**
  - http://repo.postgrespro.ru/doc/pgprosm/14.7.1/en/html
  - http://repo.postgrespro.ru/pgprosm-14
  - https://github.com/pkonotopov/shardman-docker

PGConf.Russia 2023

PostgresPro

Q&A

Спасибо!