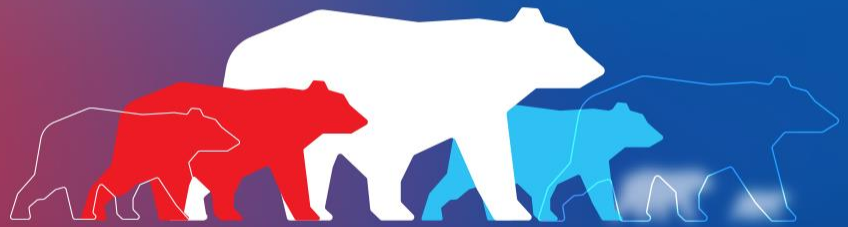
Shardman — постгрес для кластеров. Что есть сейчас, что будет завтра

Андрей Лепихов,
Postgres Professional

HL++

HighLoad++ Siberia 2019

Профессиональная конференция для разработчиков высоконагруженных систем



О себе

- Ph.D. в параллельных системах баз данных
- Core Developer в Postgres Professional

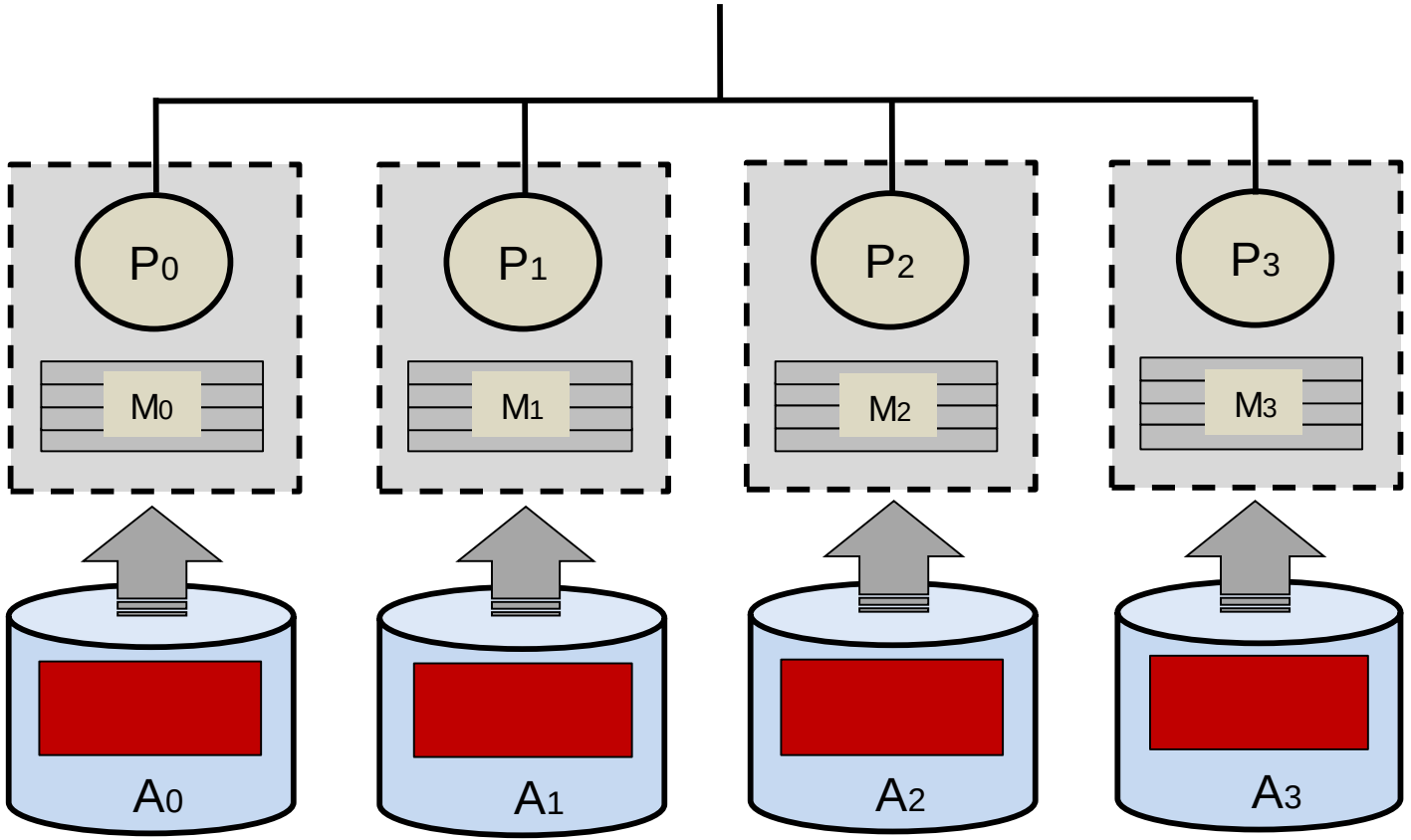
Postgres-специализация:
- WAL
- Planner
- B-tree/GiST/SP-GiST access method
- VACUUM

Кластер — для чего?

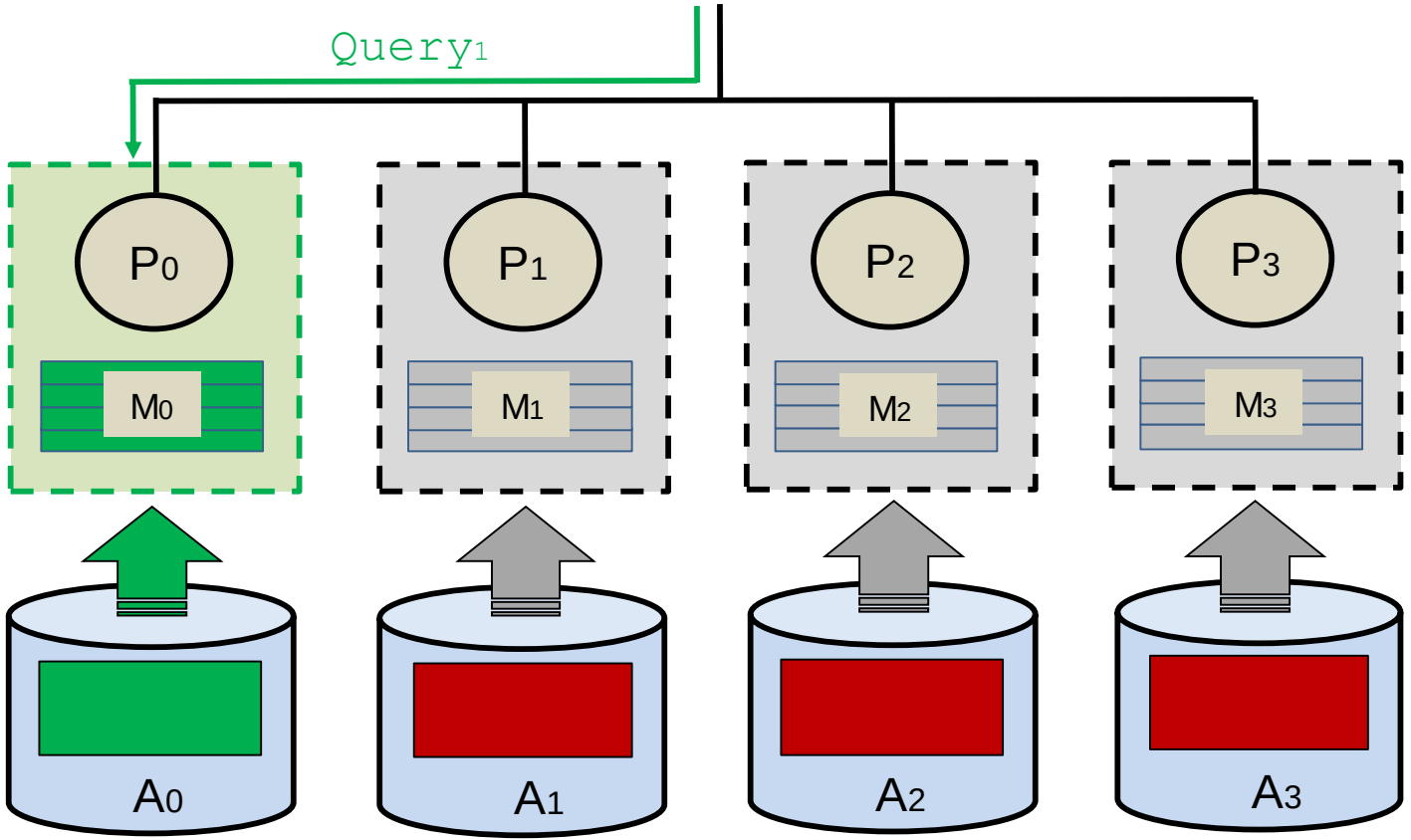
- Шардирование БД размером от 10 ТБ
- Доступность данных
- Масштабирование

  - Ускорение (*увеличить TPS*)
  - Расширяемость (*сохранить TPS на растущей БД*)

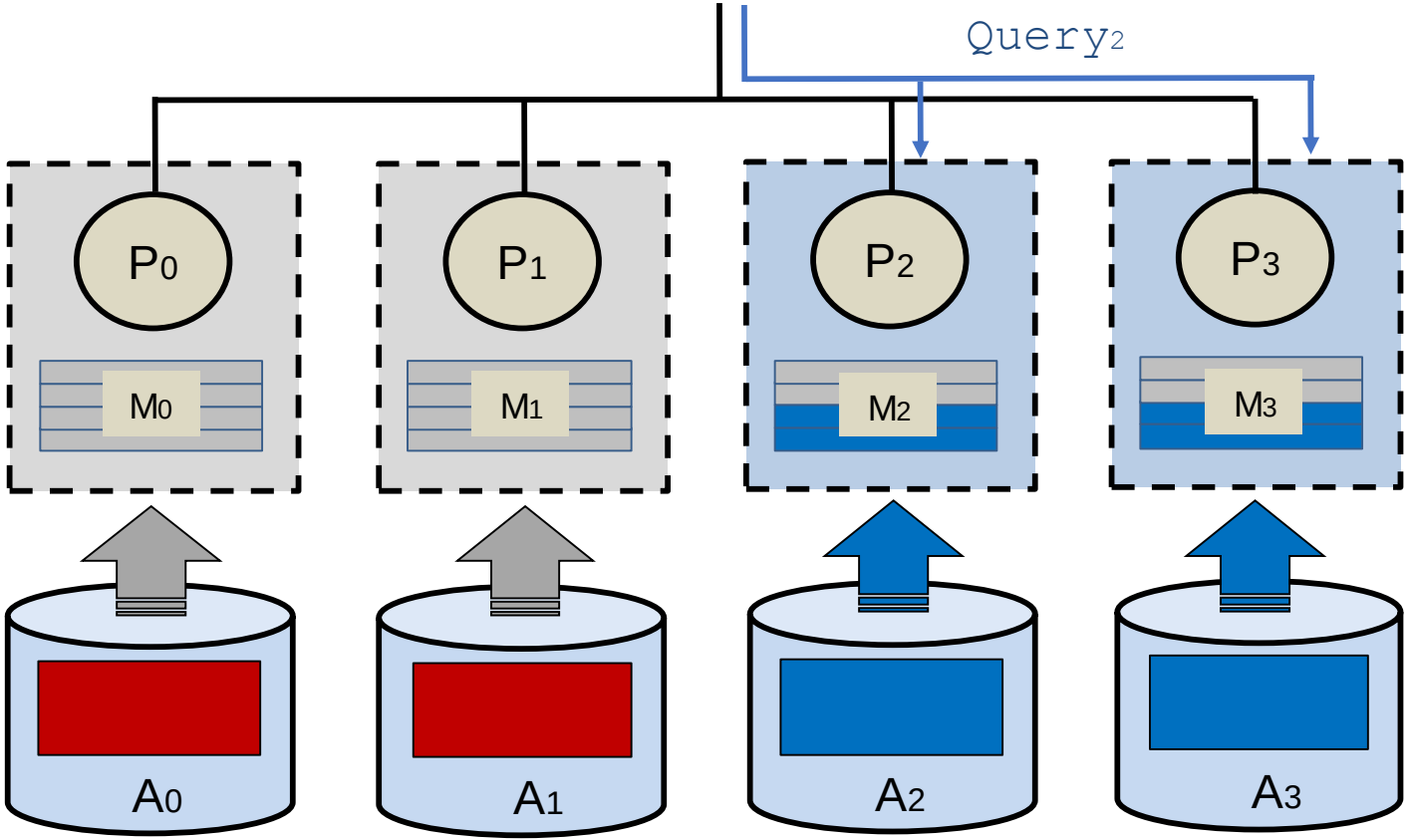
Утилизация кластера



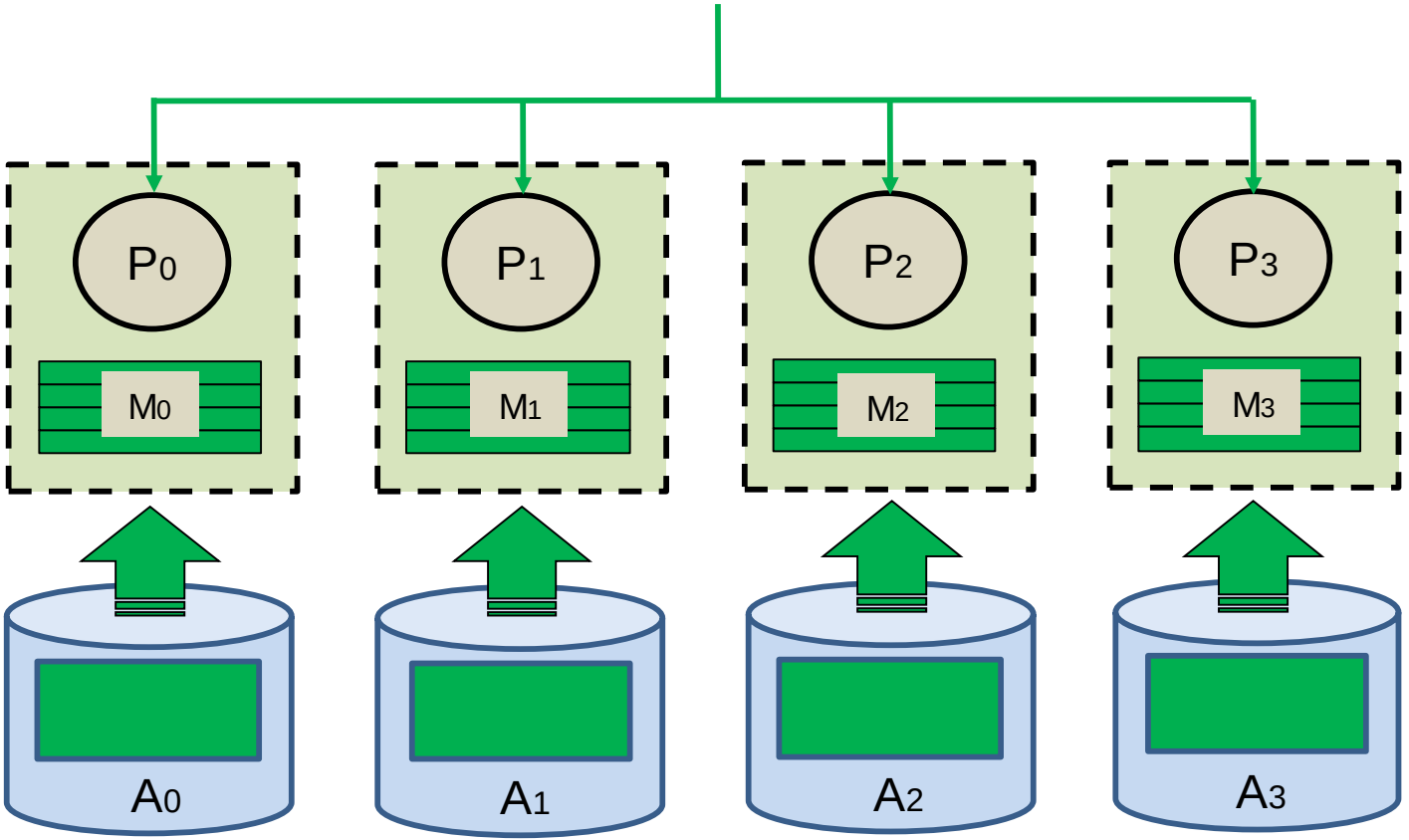
Утилизация кластера: OLTP



Утилизация кластера: OLTP



Утилизация кластера: OLAP



Задачи

- Планирование запроса
  — JOIN
  — Агрегаты
  — Сортировки
- Доступность данных
- Распределенные транзакции
- Балансировка загрузки

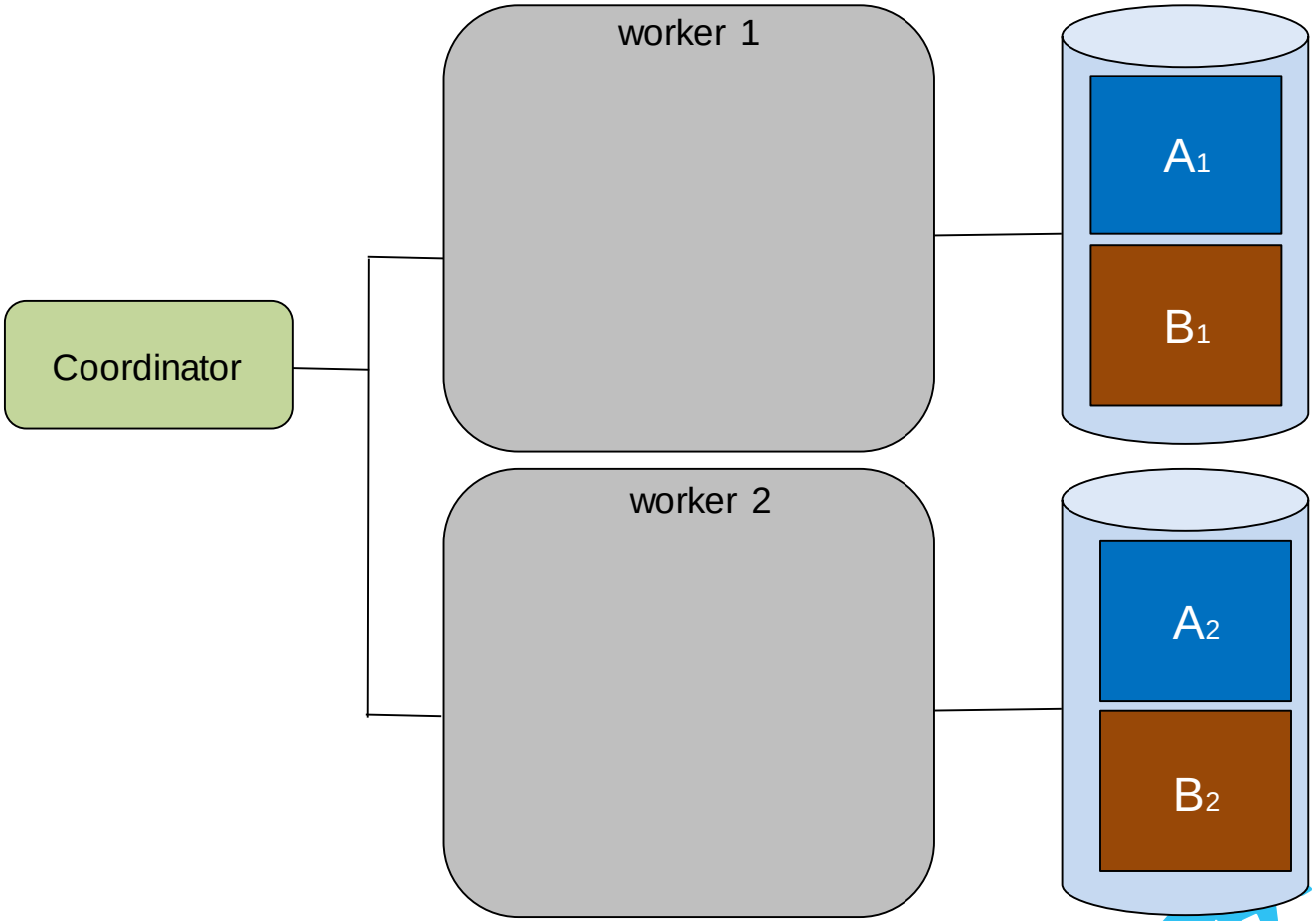
Альтернативы

- Citus
https://www.citusdata.com/
- Postgres-XL
https://www.postgres-xl.org/
- Greenplum
https://greenplum.org/

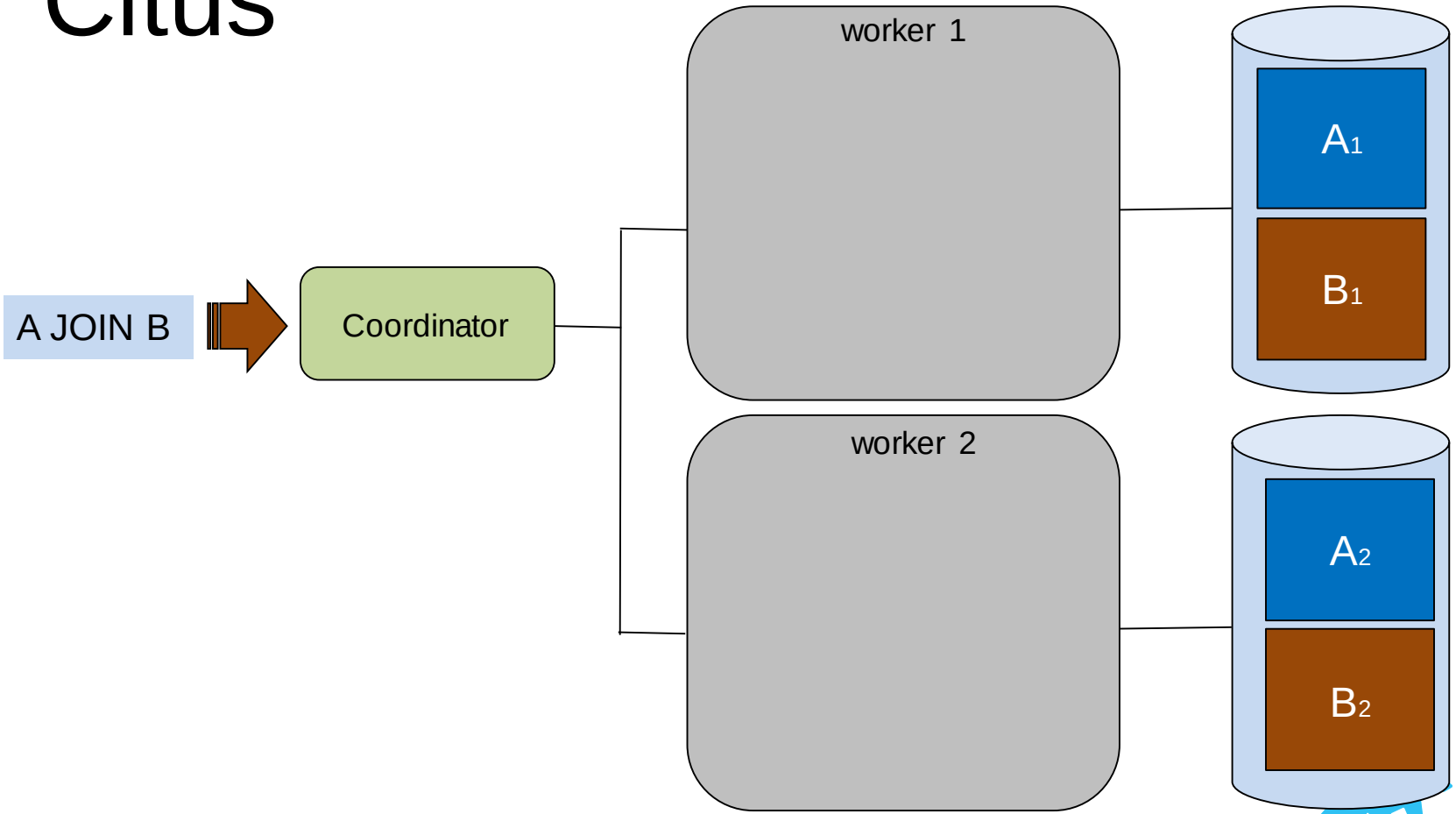
Citus

- Чистый EXTENSION
- Преобразует запрос в серии подзапросов, которые могут быть выполнены независимо каждым инстансом
- Динамически перераспределяет данные с материализацией между сериями подзапросов
- Собственный планнер
- Собственный механизм шардирования и прунинга
- Выделенный координатор

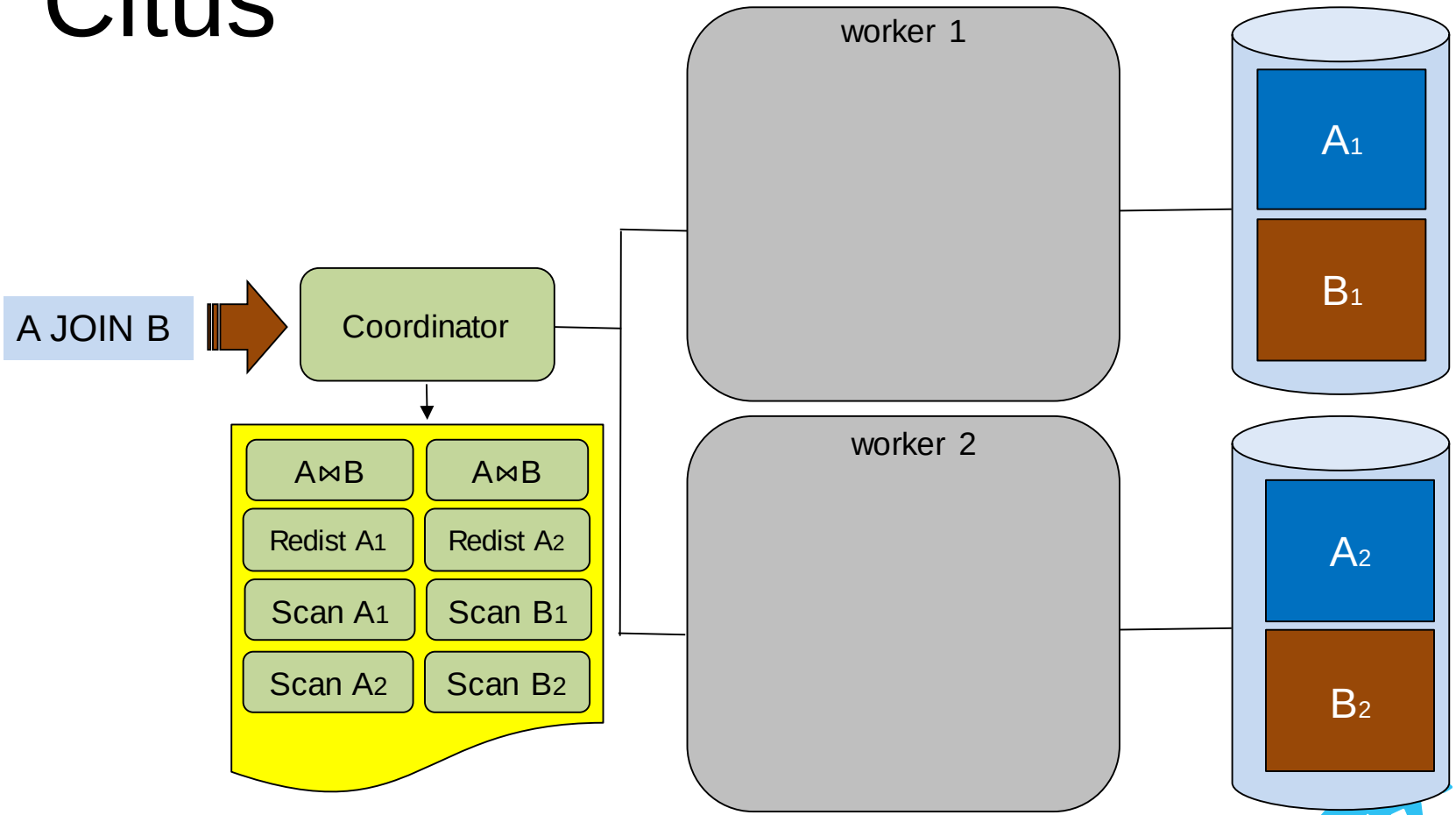
Citus



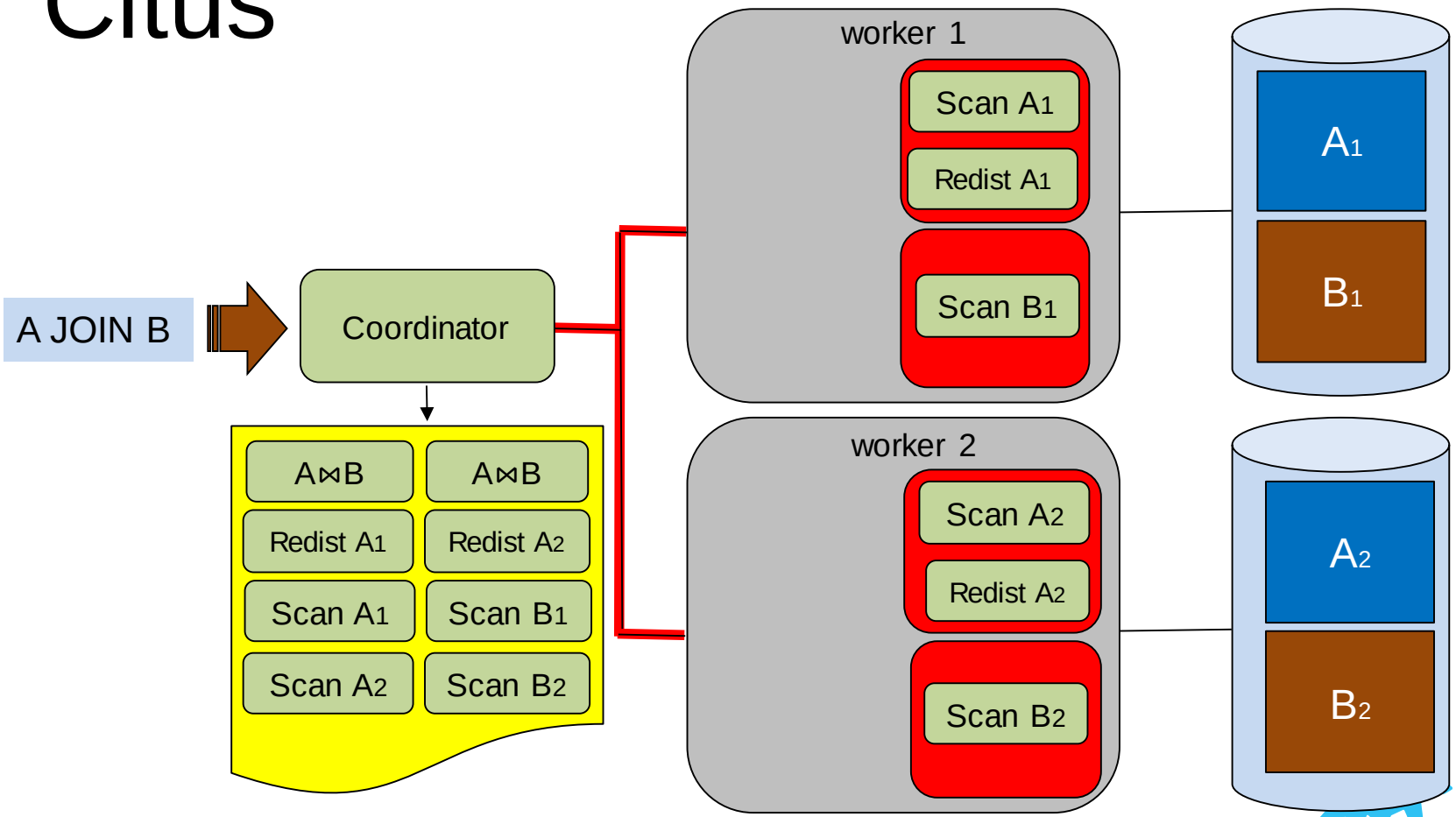
Citus



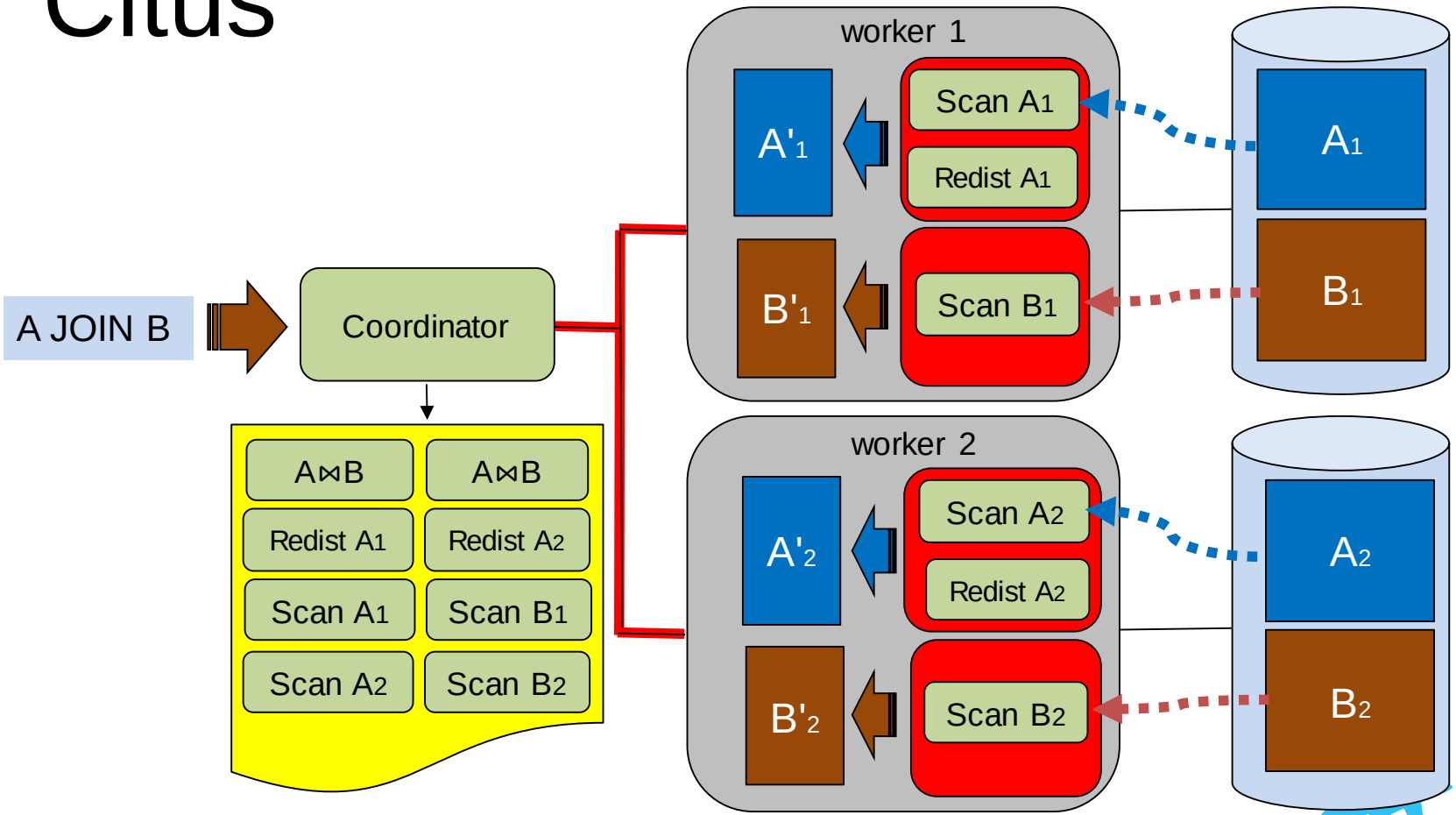
Citus



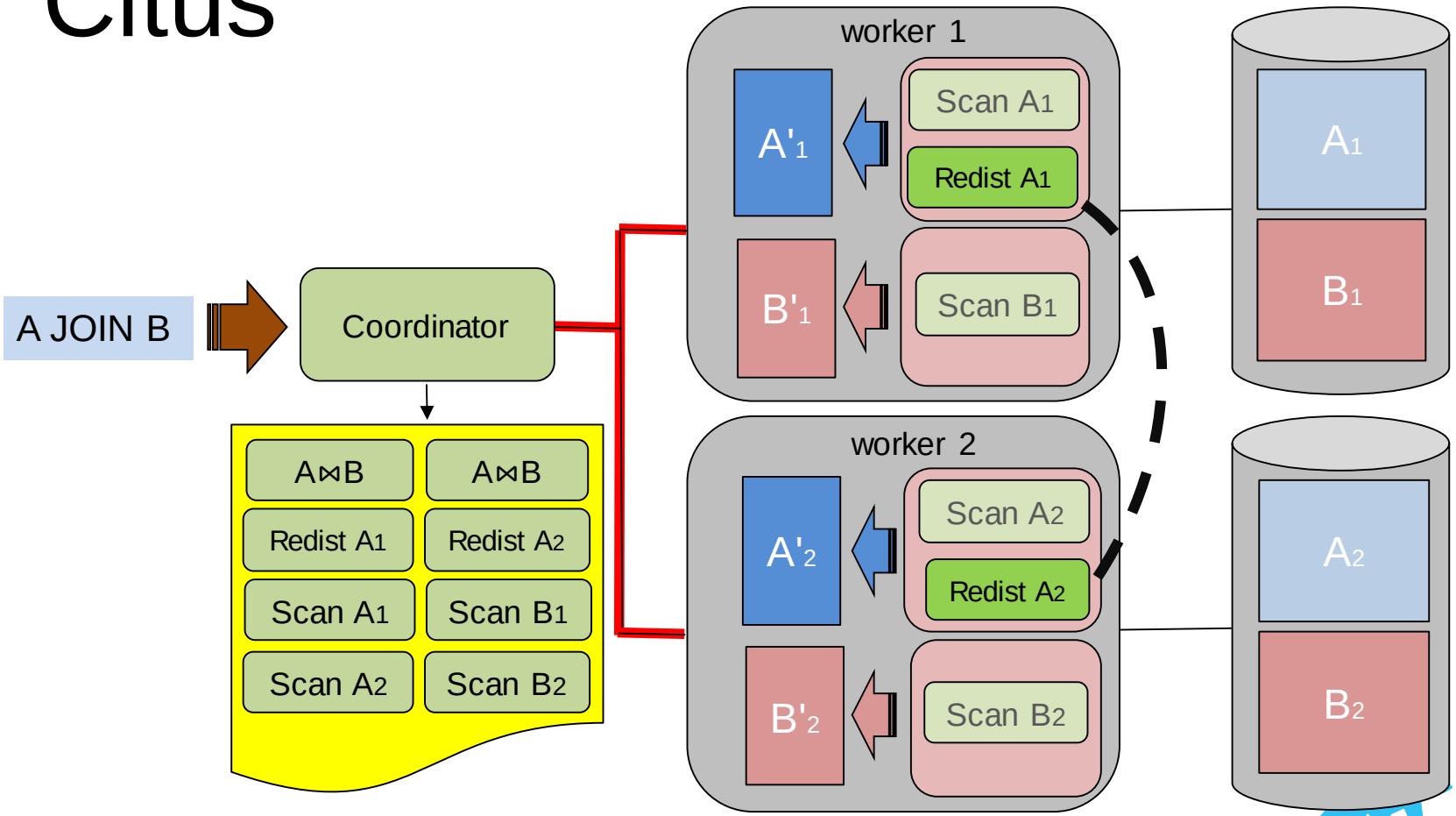
Citus



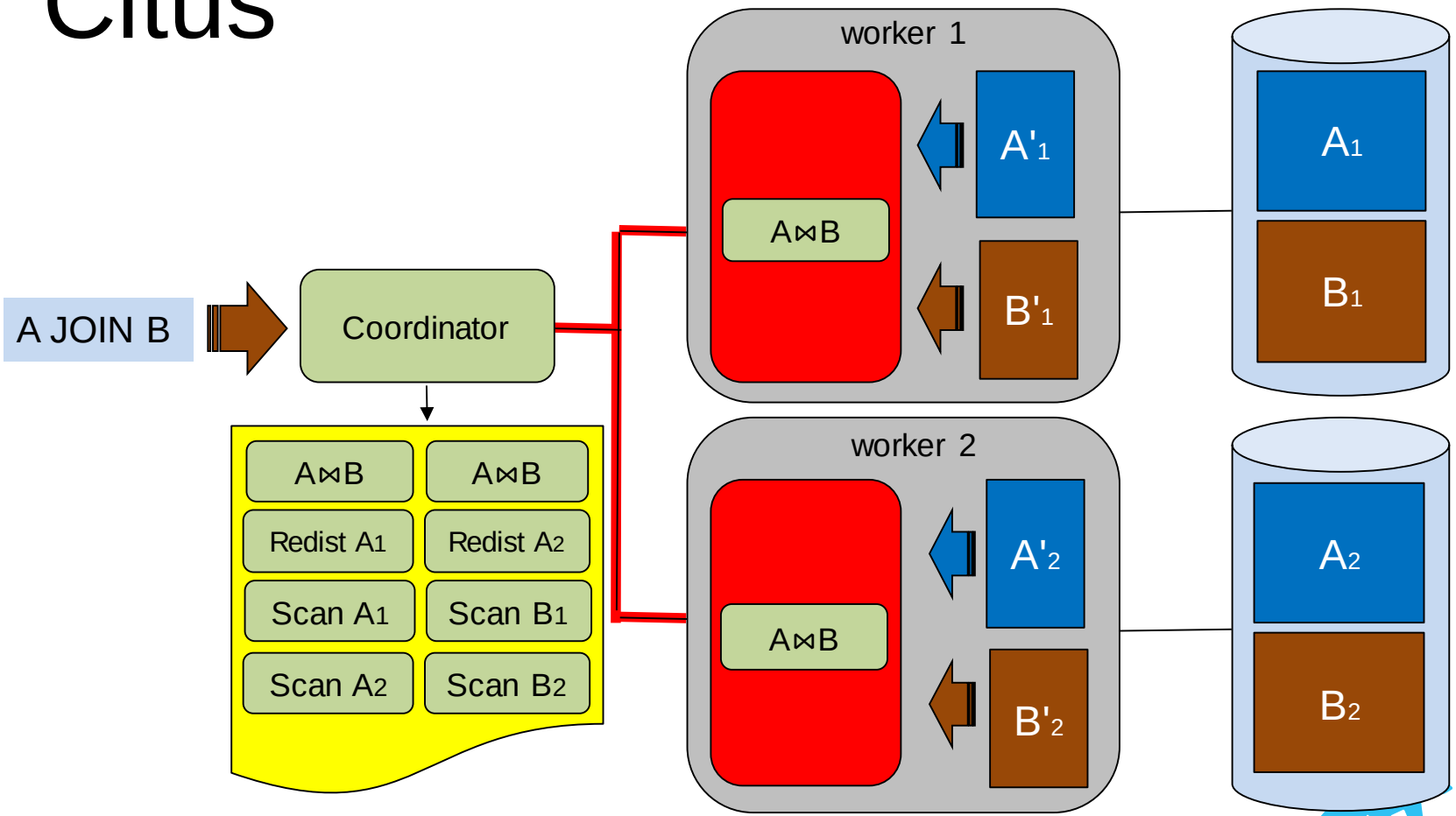
Citus



Citus



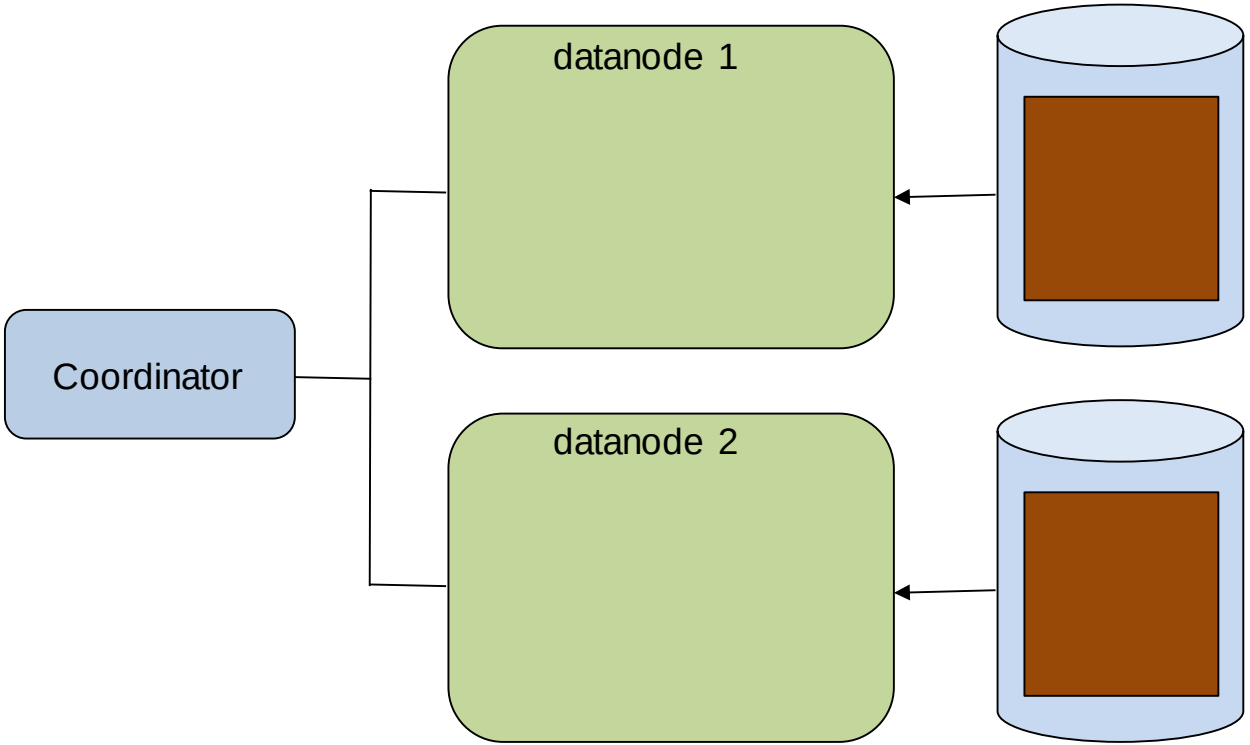
Citus



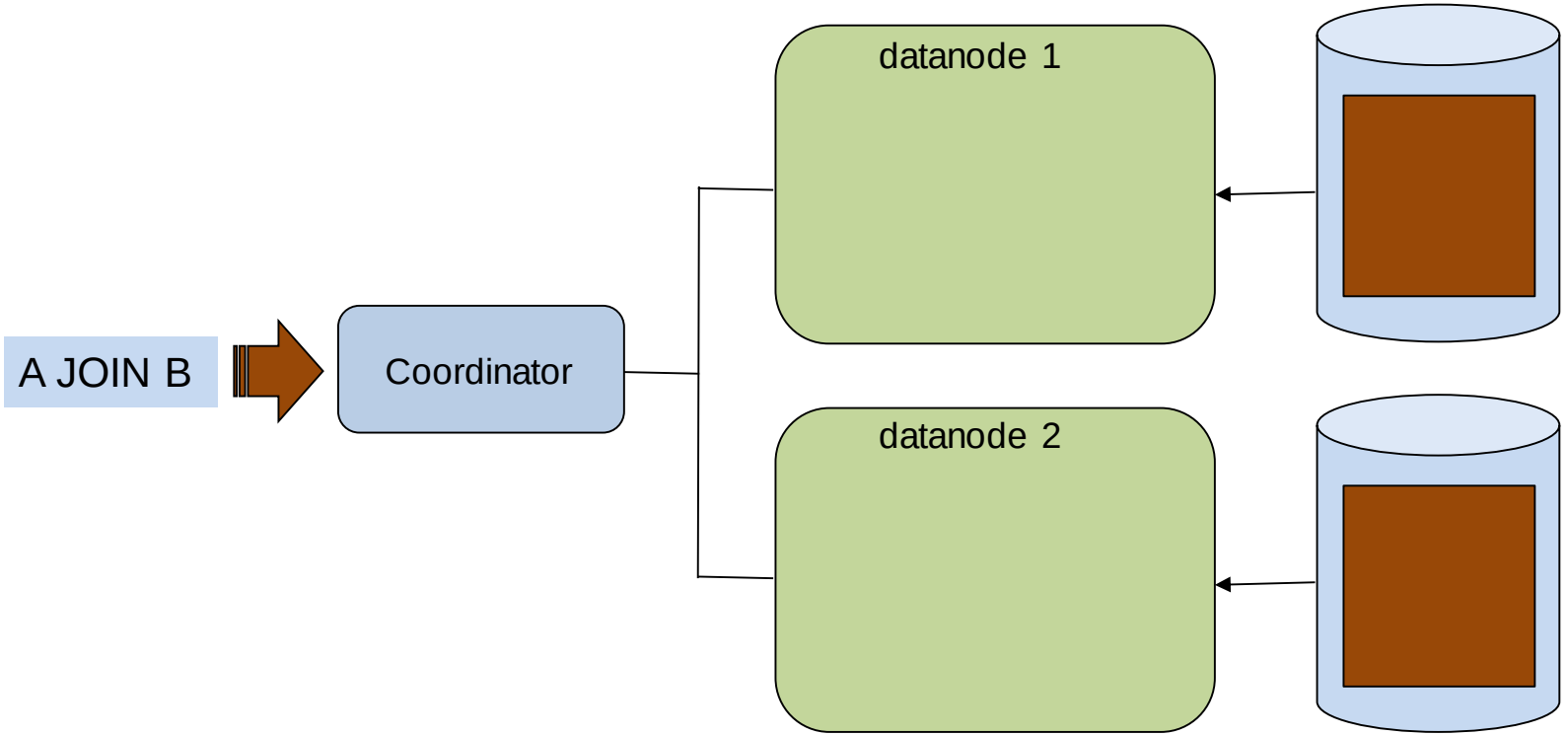
Postgres-XL

- Строит план один раз
- Разрезает план в местах, где требуется перераспределение данных
- Нет жесткого разделения на координатор и воркер
- Собственный планнер
- Собственный механизм шардирования и прунинга

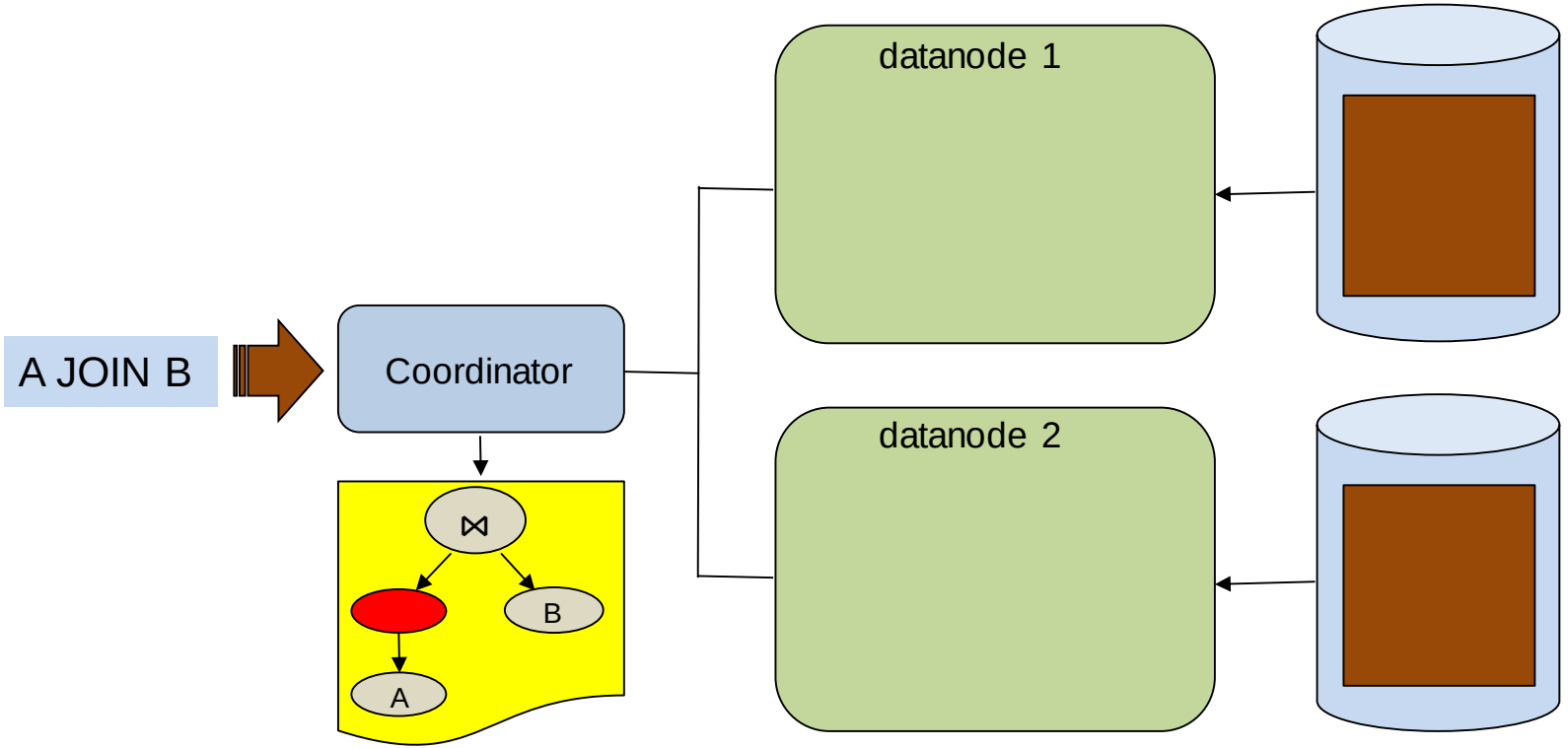
Postgres-XL



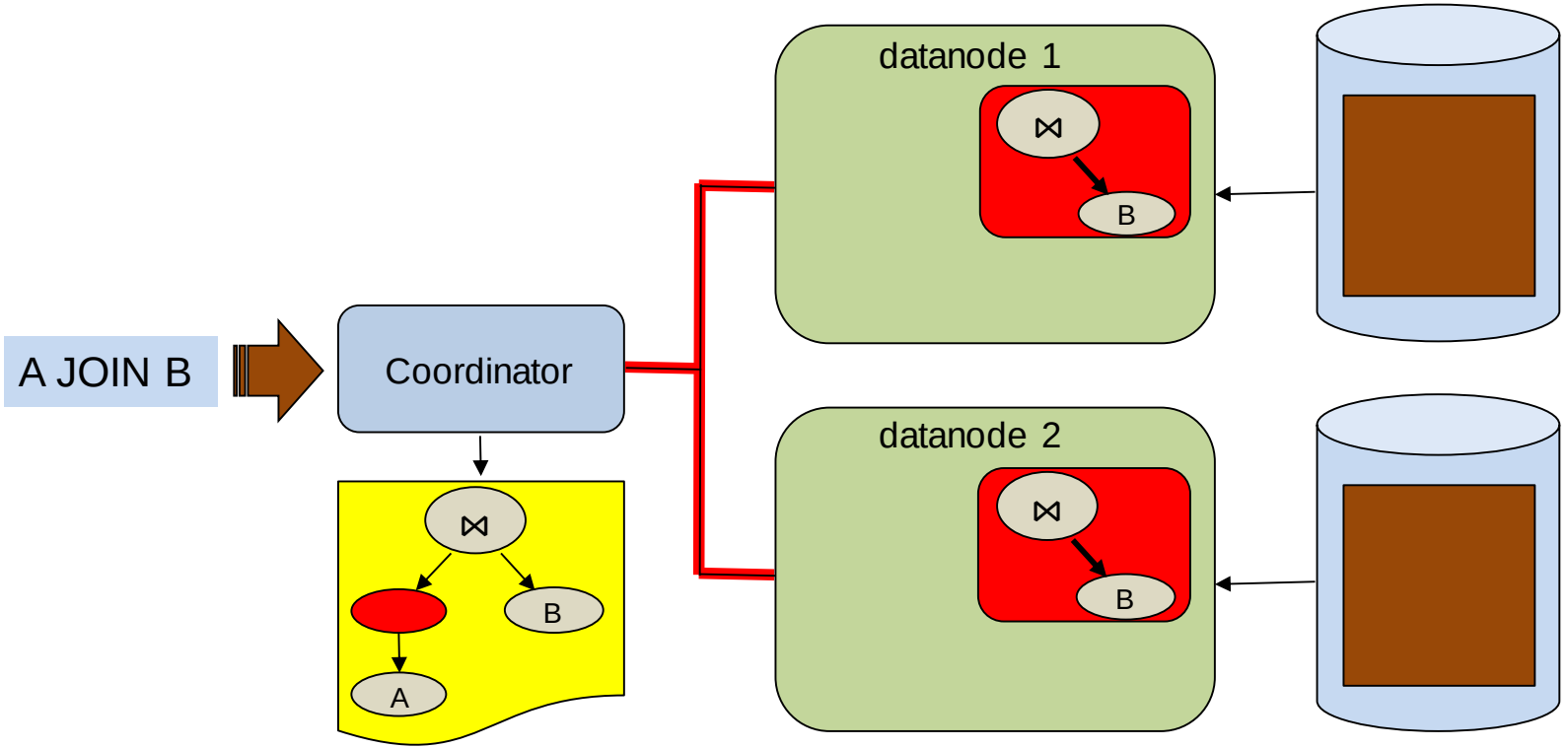
Postgres-XL



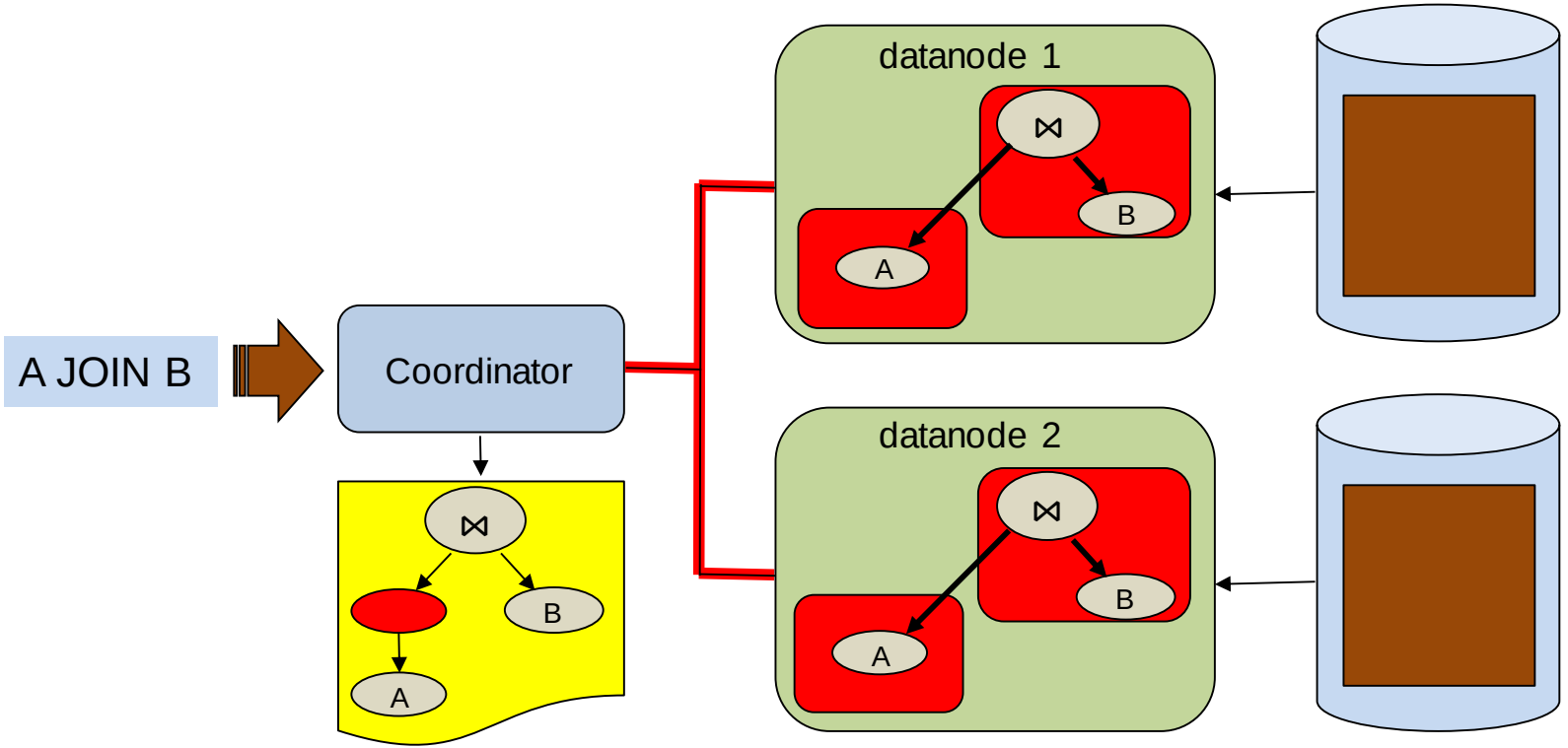
Postgres-XL



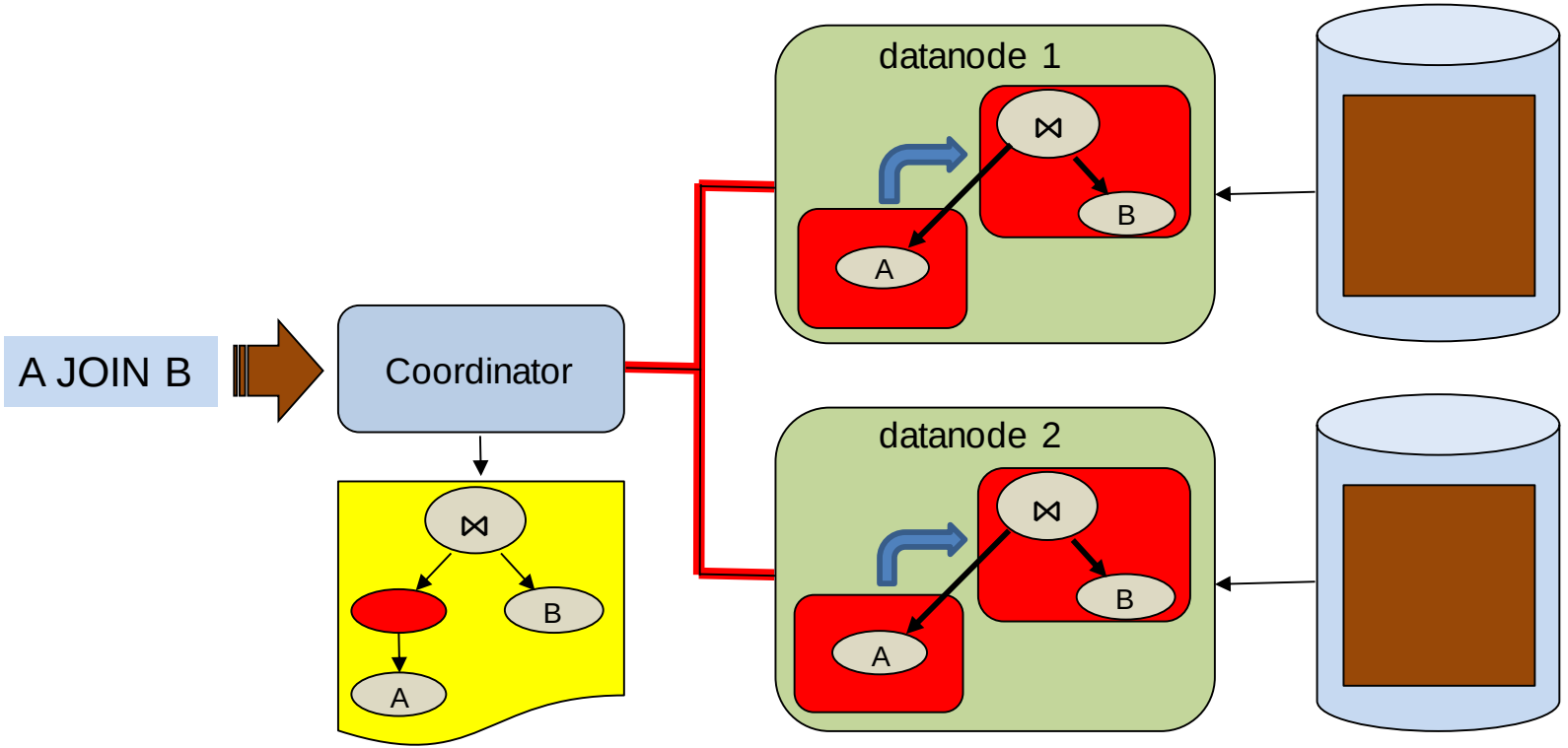
Postgres-XL



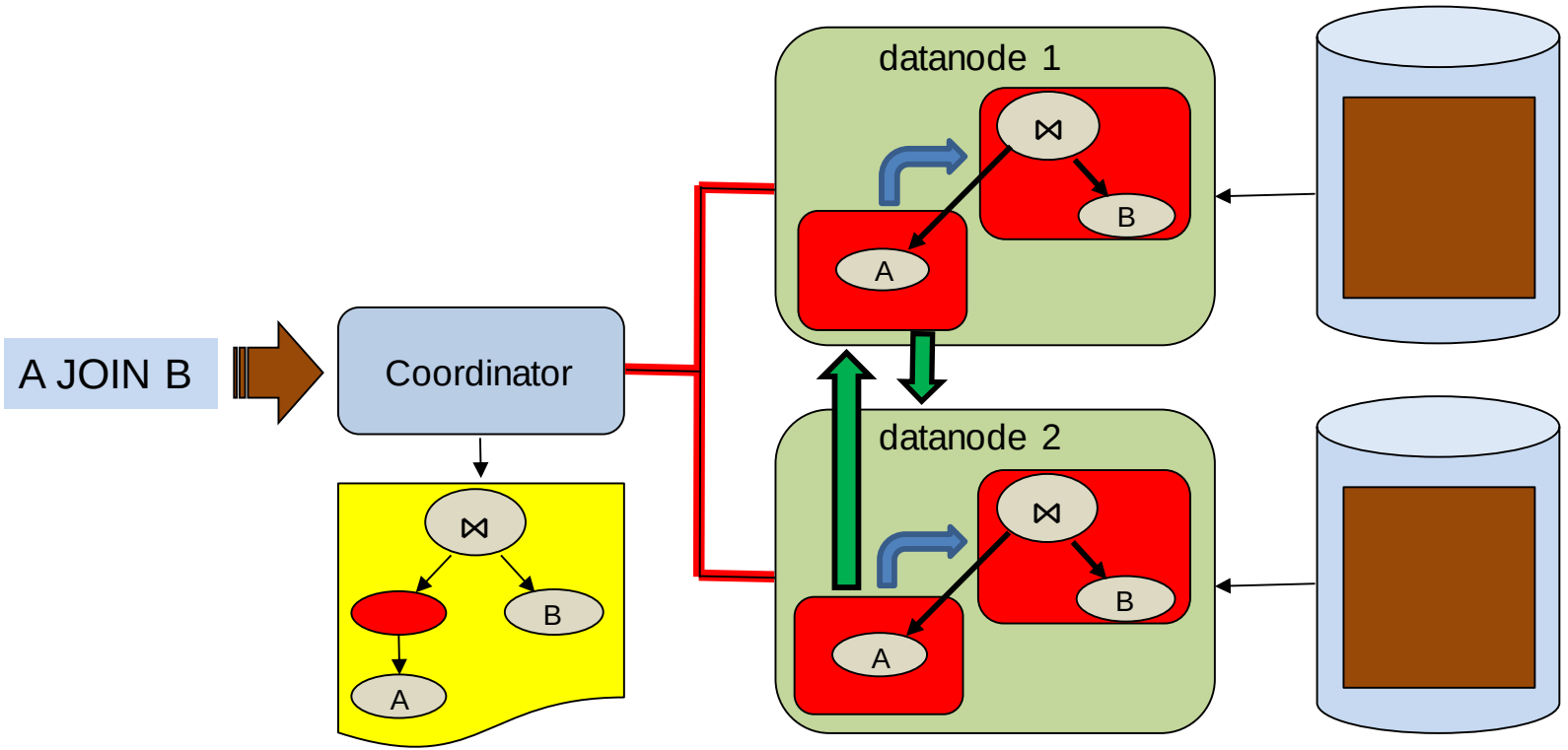
Postgres-XL



Postgres-XL



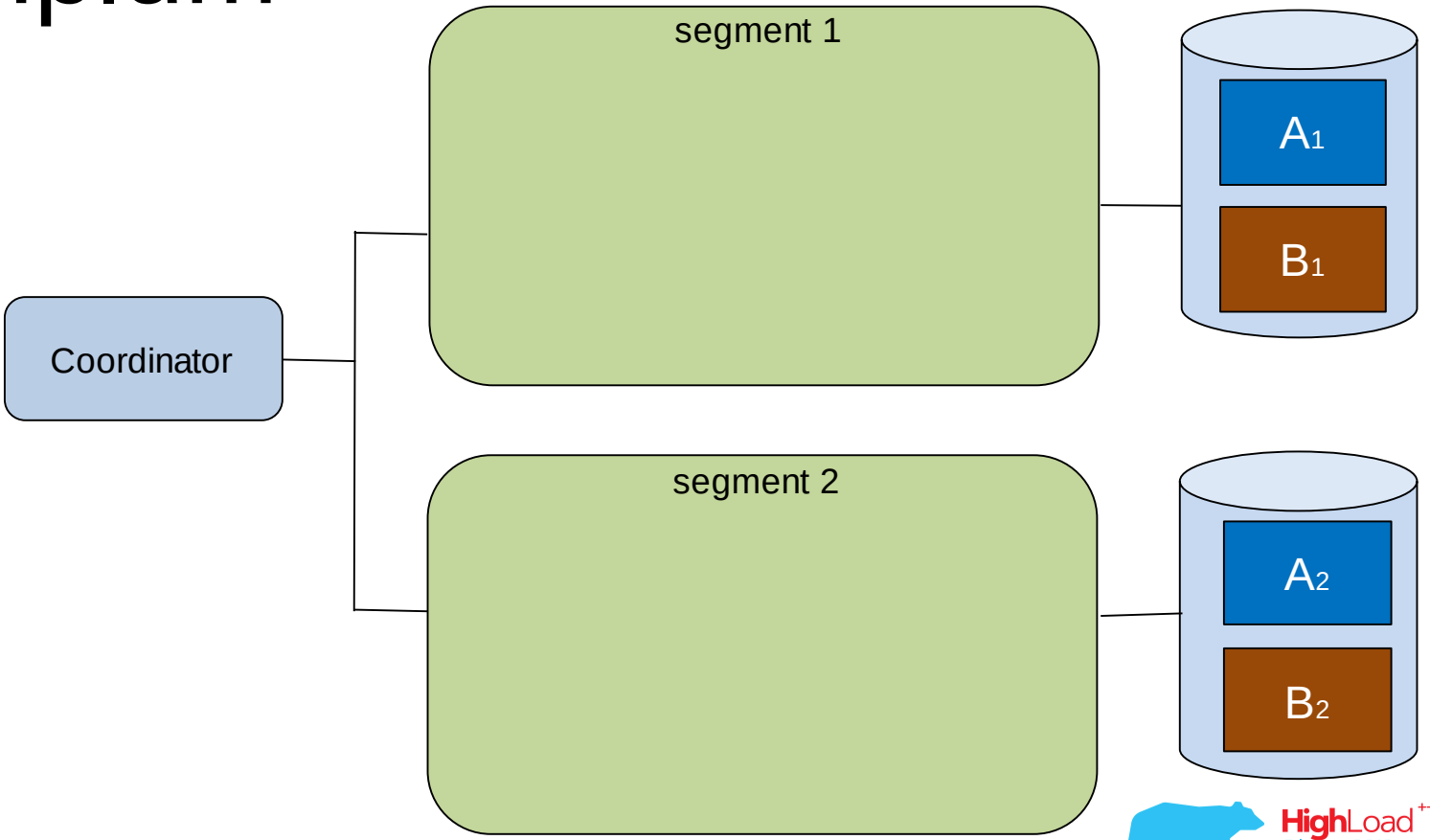
Postgres-XL



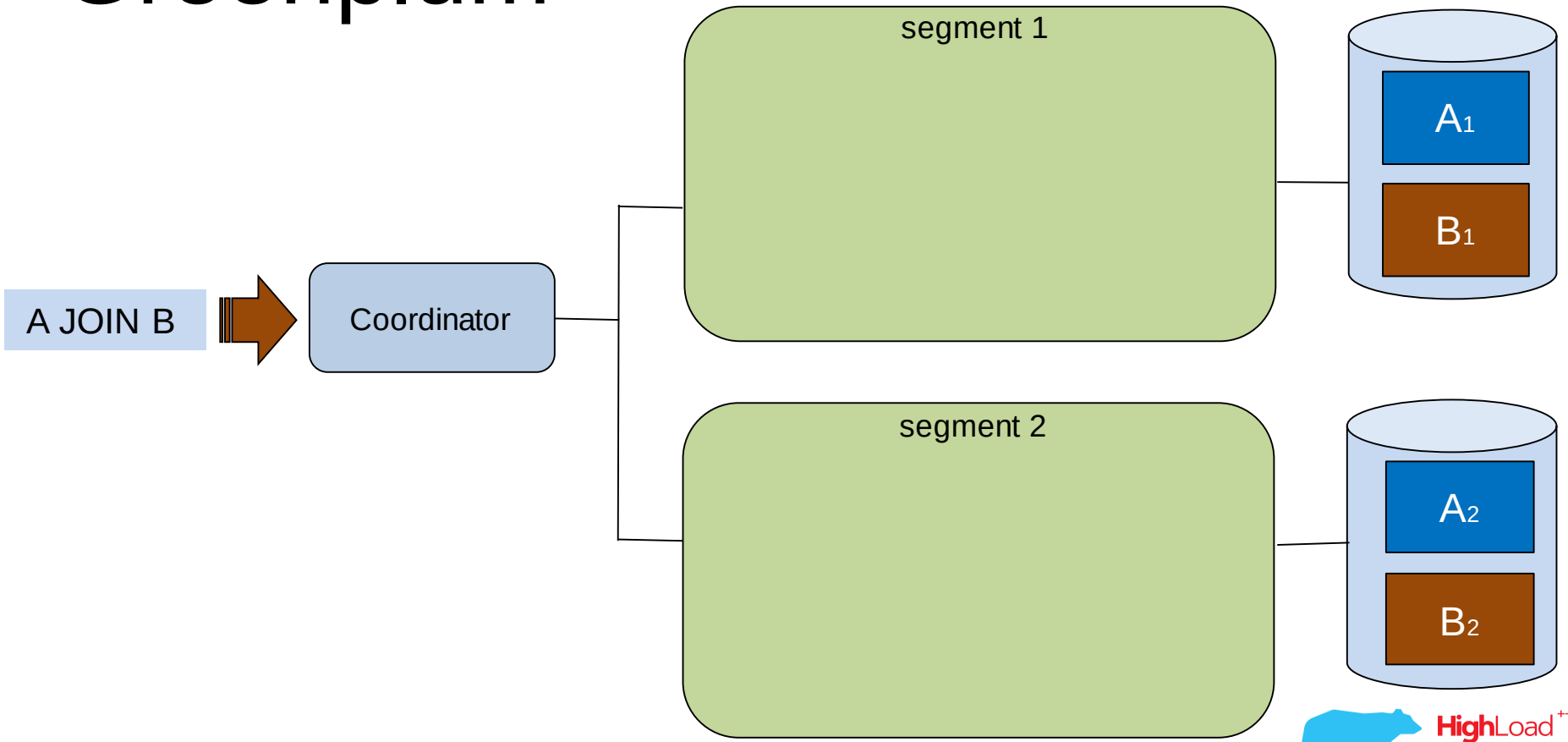
Greenplum

- Строит план один раз
- Делит план горизонтально (в целях перераспределения данных) и вертикально (подпланы, которые могут быть выполнены параллельно)
- Собственный планнер
- Собственный механизм шардирования и прунинга

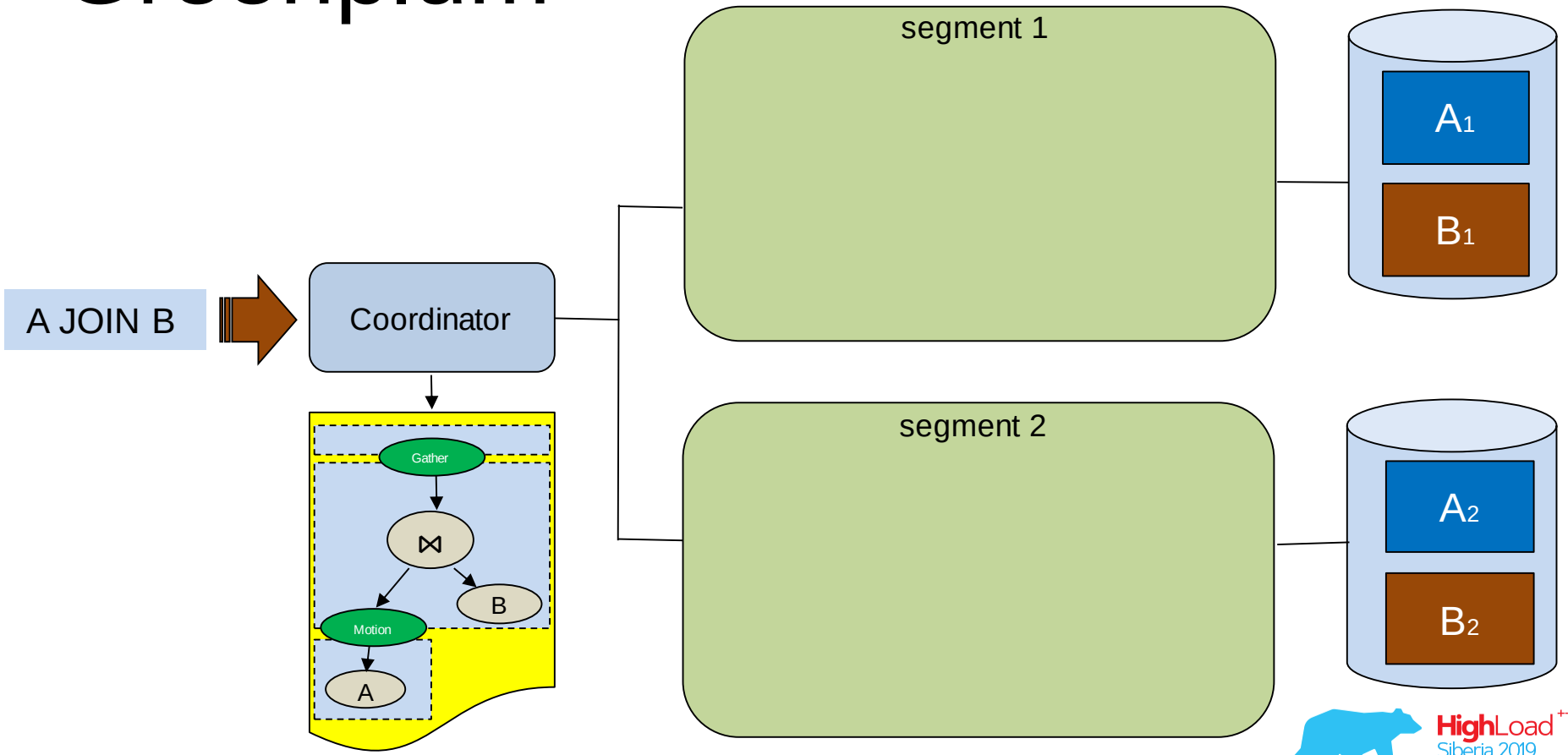
Greenplum



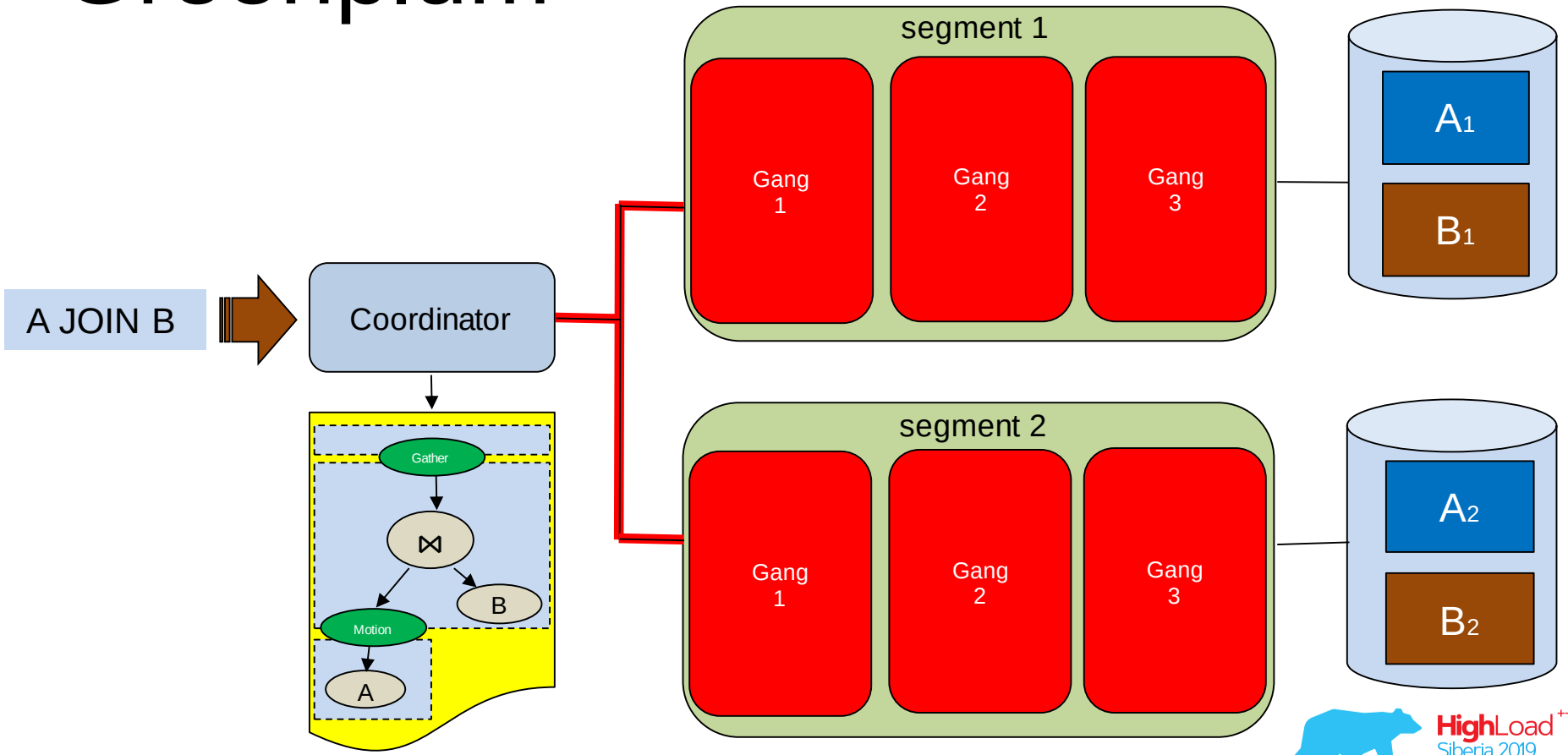
Greenplum



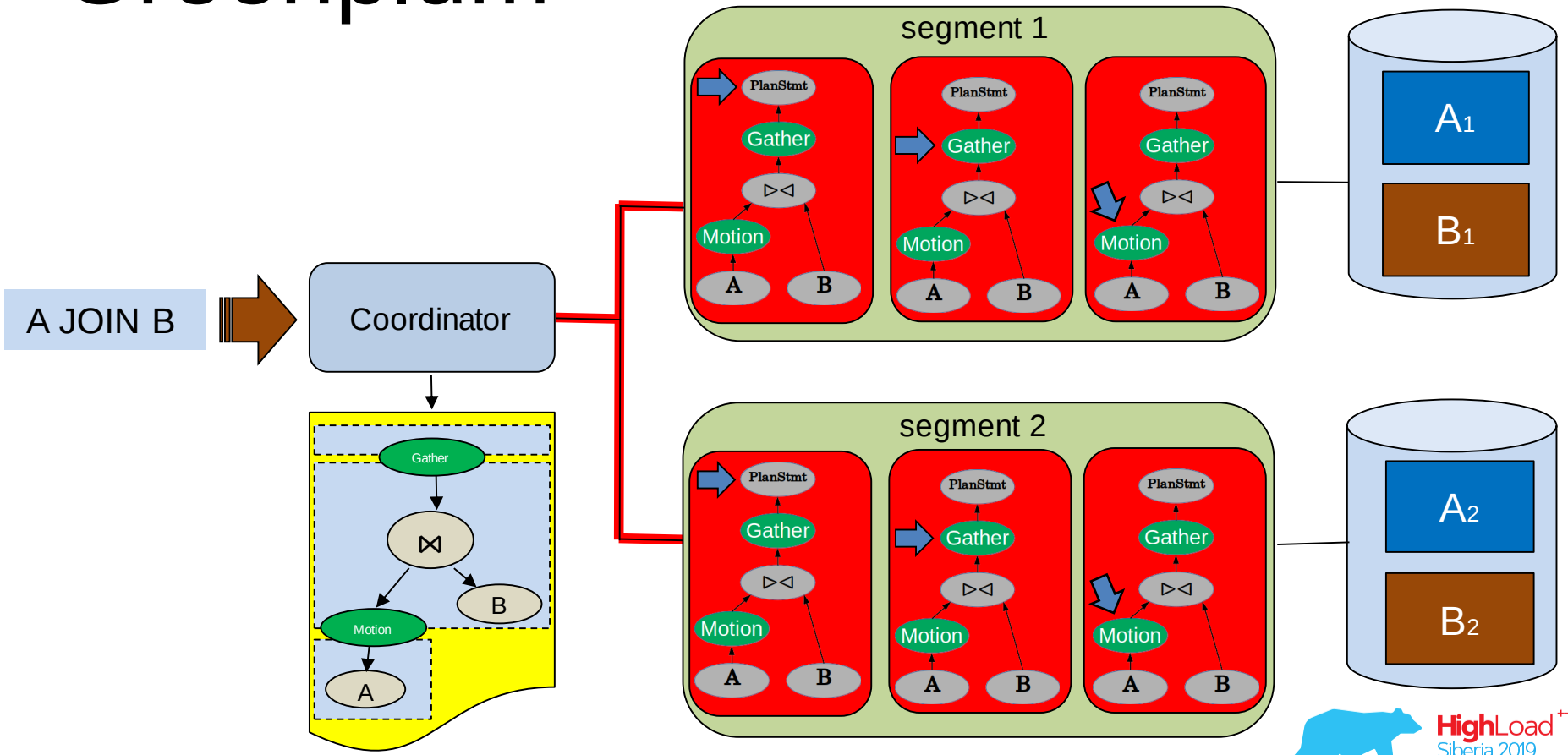
Greenplum



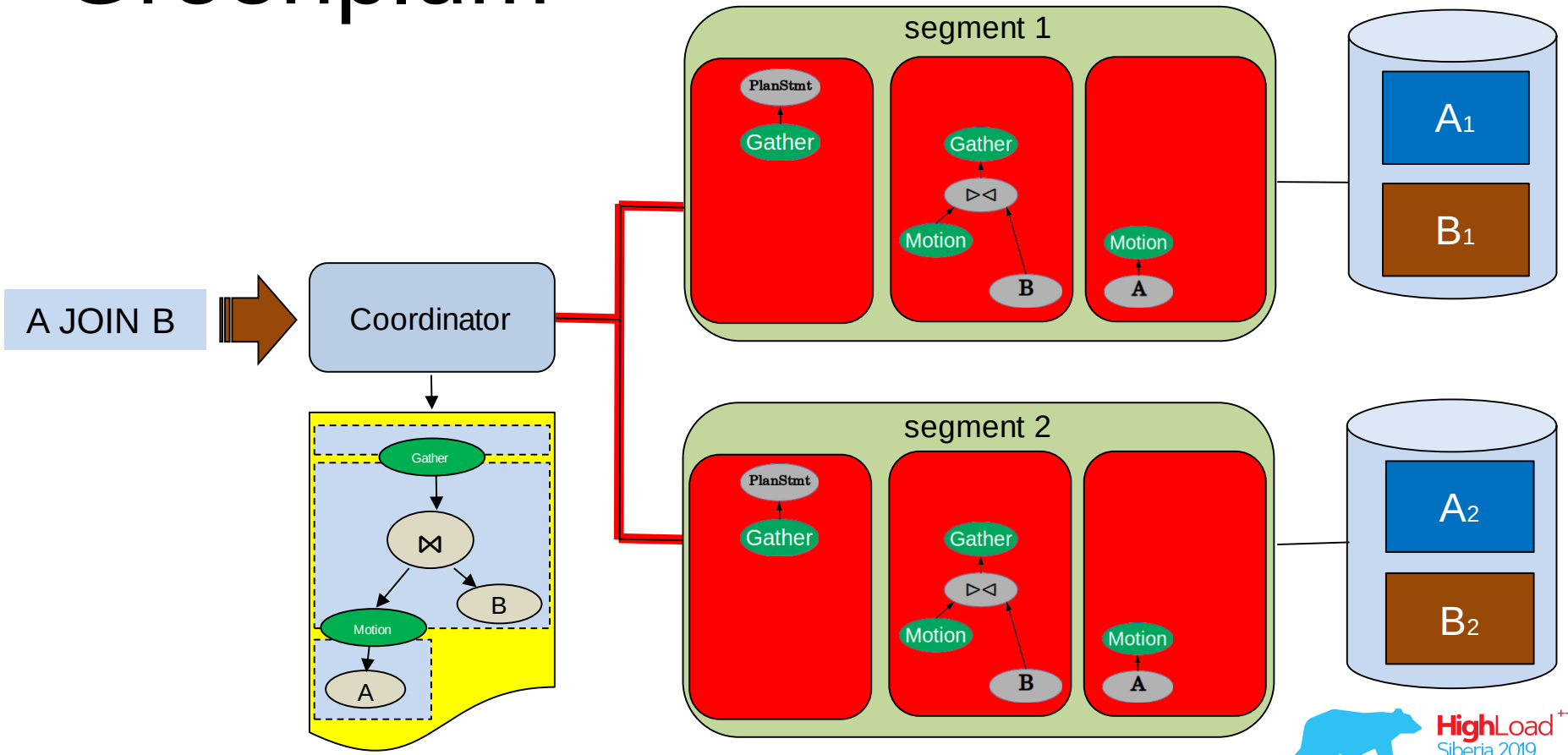
Greenplum



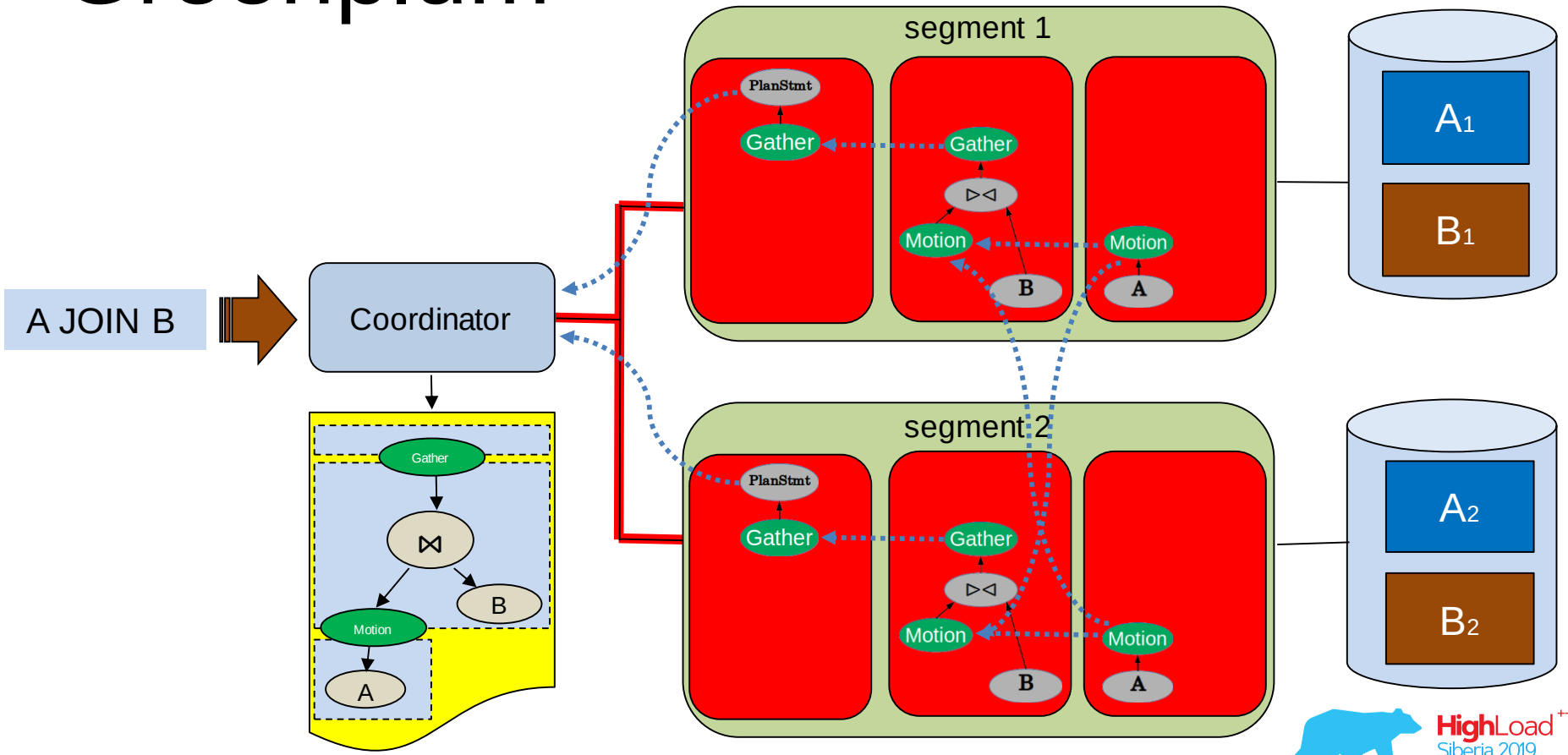
Greenplum



Greenplum



Greenplum



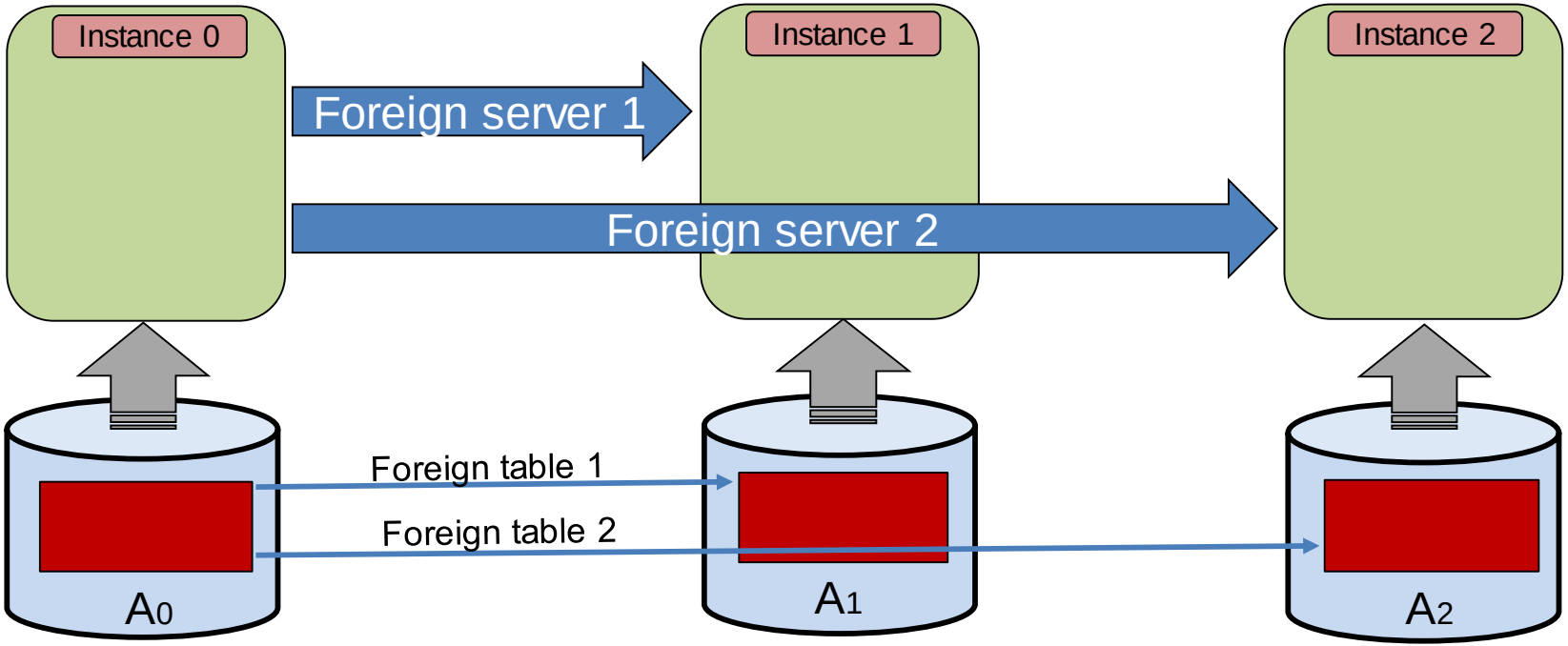
Итого

- Акцент на OLTP либо OLAP
- Высокая трудоёмкость апгрейда до новой версии ванильного постгреса
- Невысокая вероятность поддержки сообществом PostgreSQL в будущем

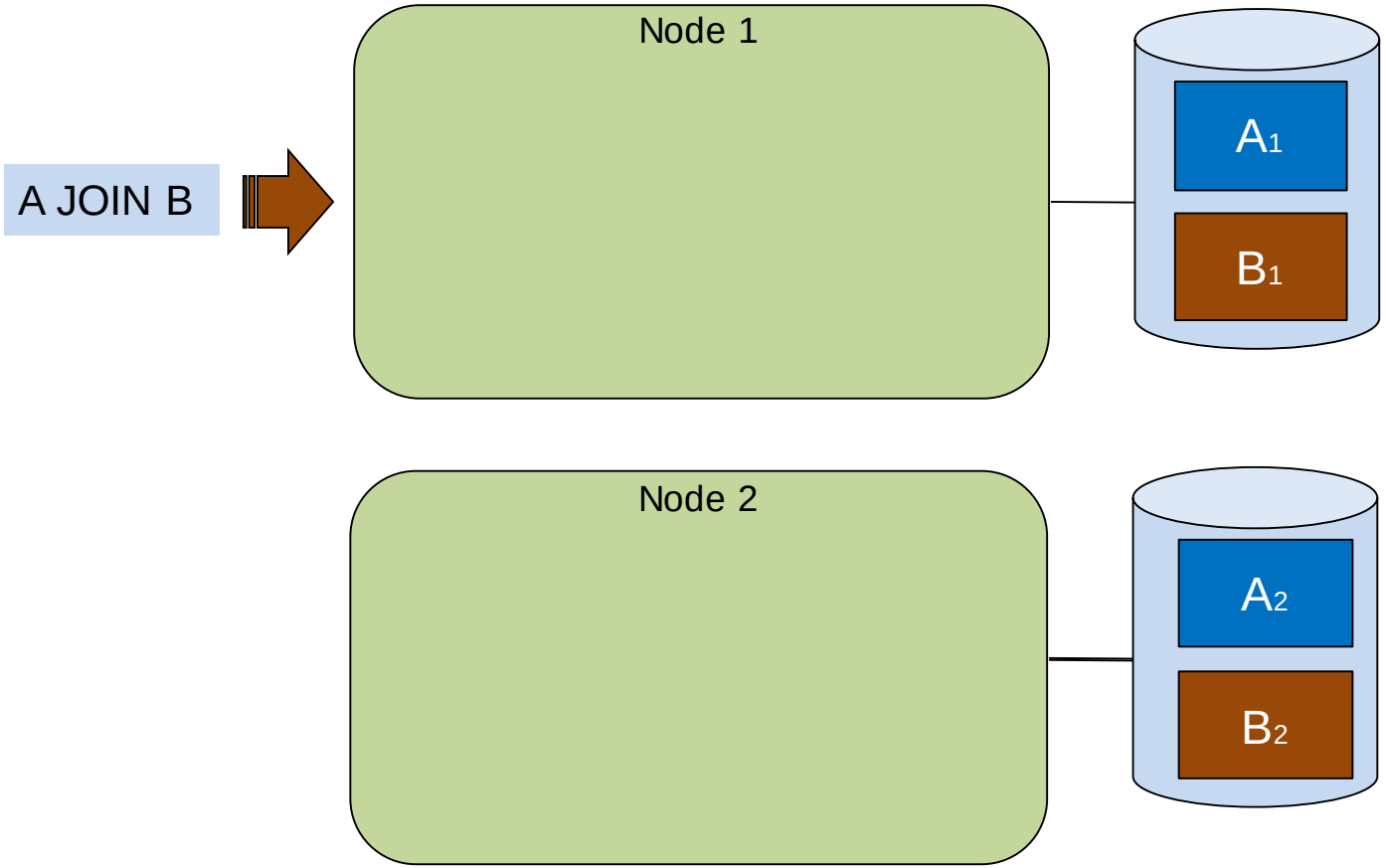
Шардман: принципы

- EXTENSION везде, где возможно
- Partitioning для шардирования и FDW для коннектов между инстансами
- Каждый инстанс может быть координатором
- Можно конфигурировать для OLTP и OLAP
- Ориентир на интеграцию в ванильный постгрес

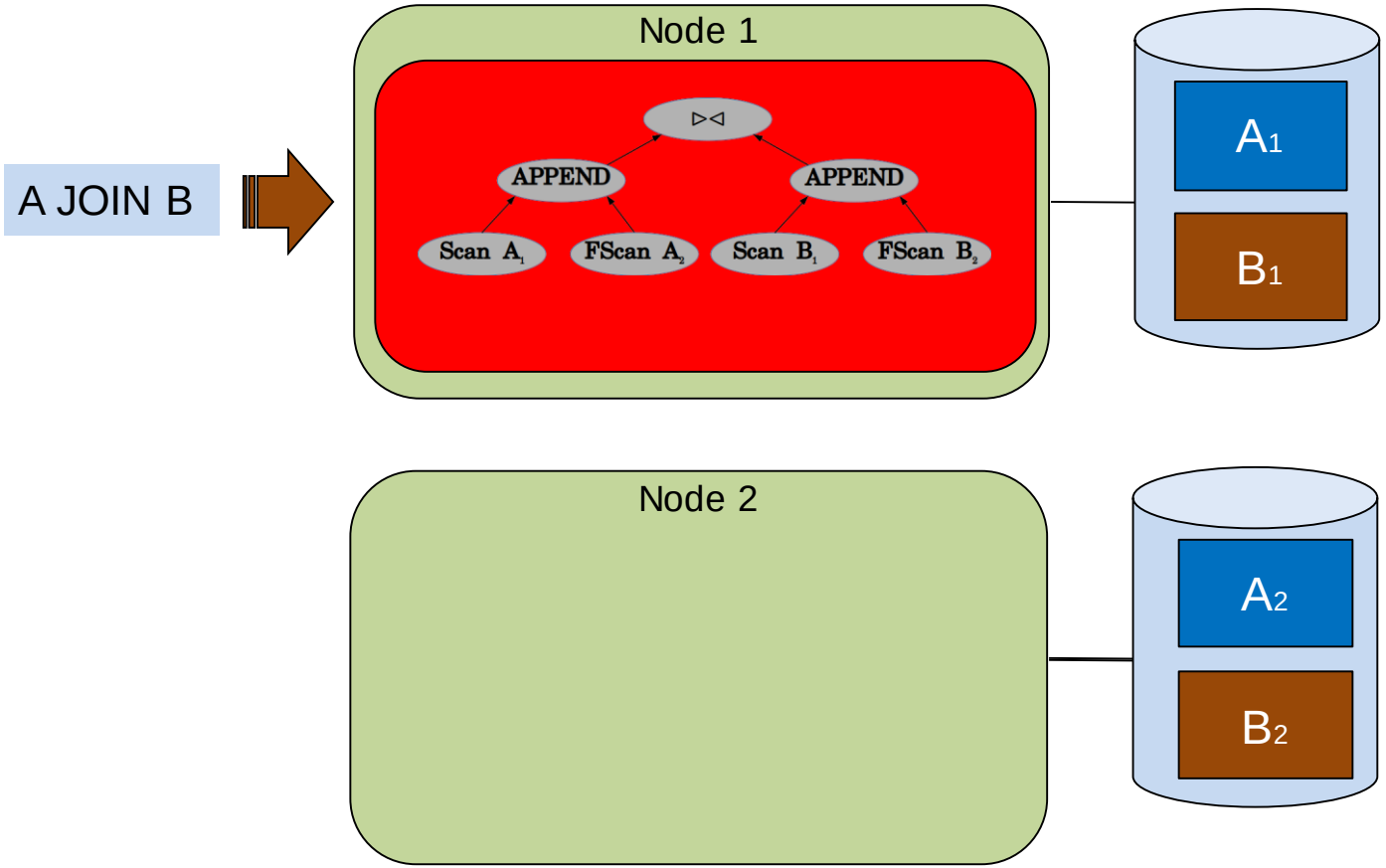
FDW + Partitioning



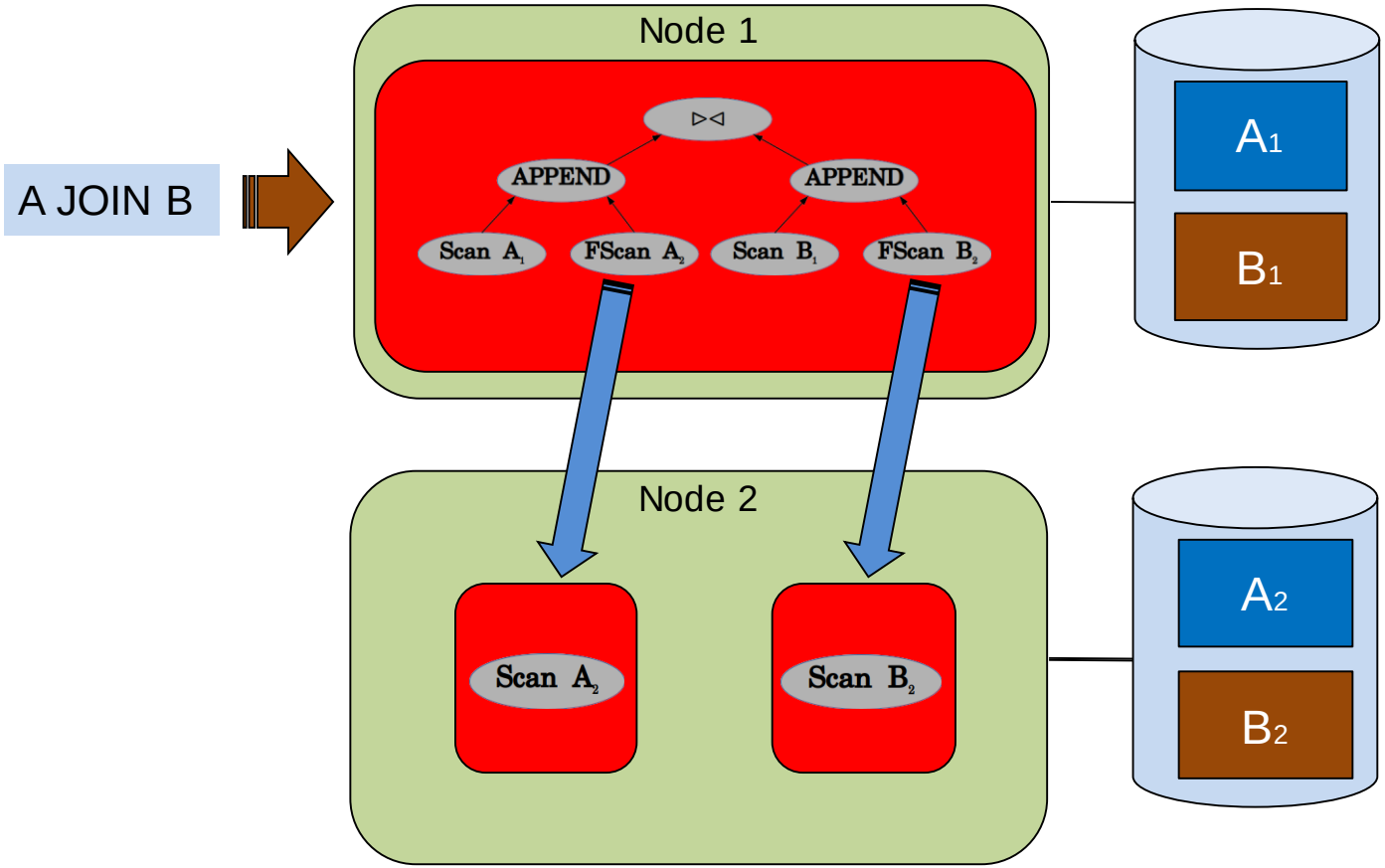
FDW + Partitioning



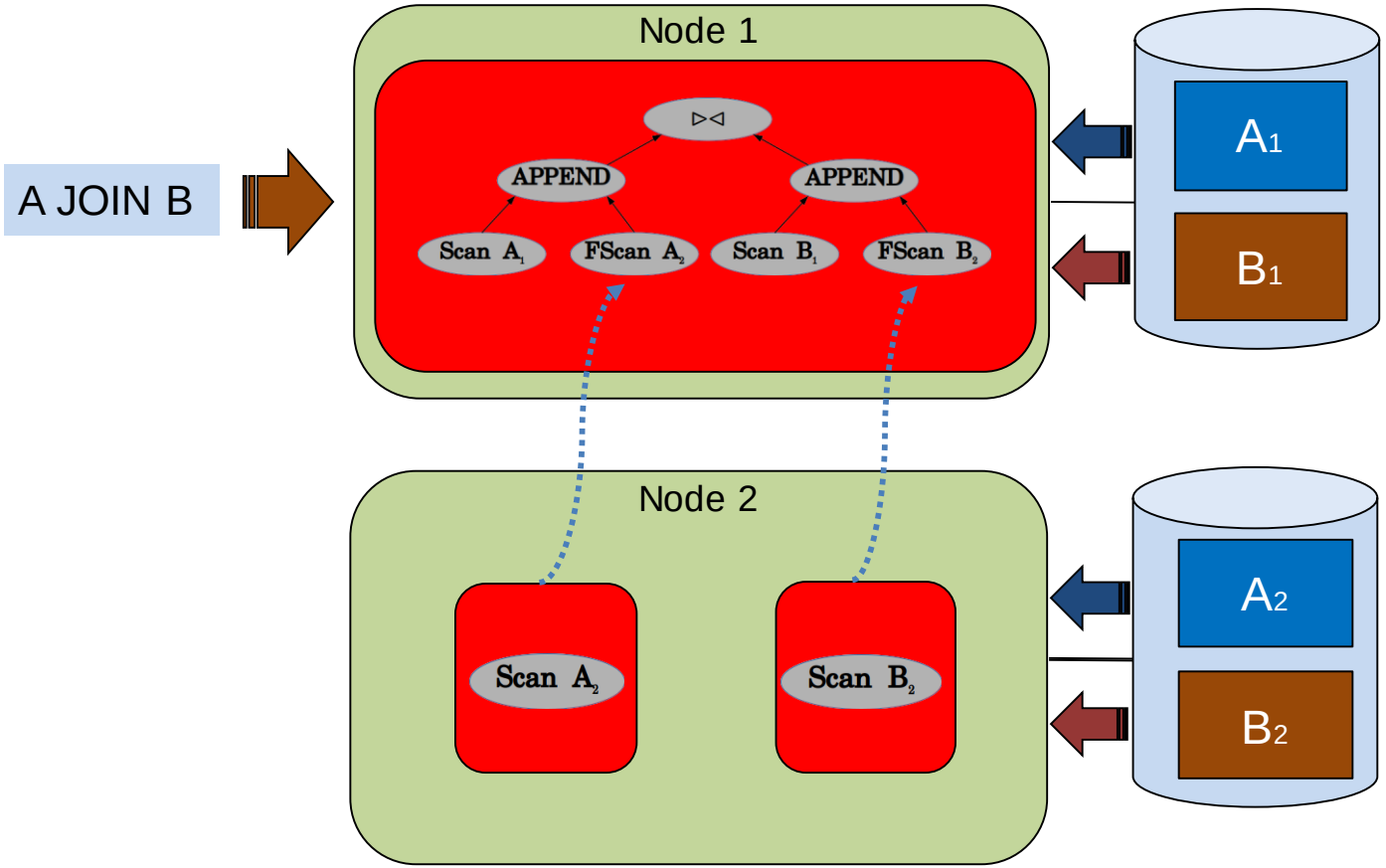
FDW + Partitioning



FDW + Partitioning



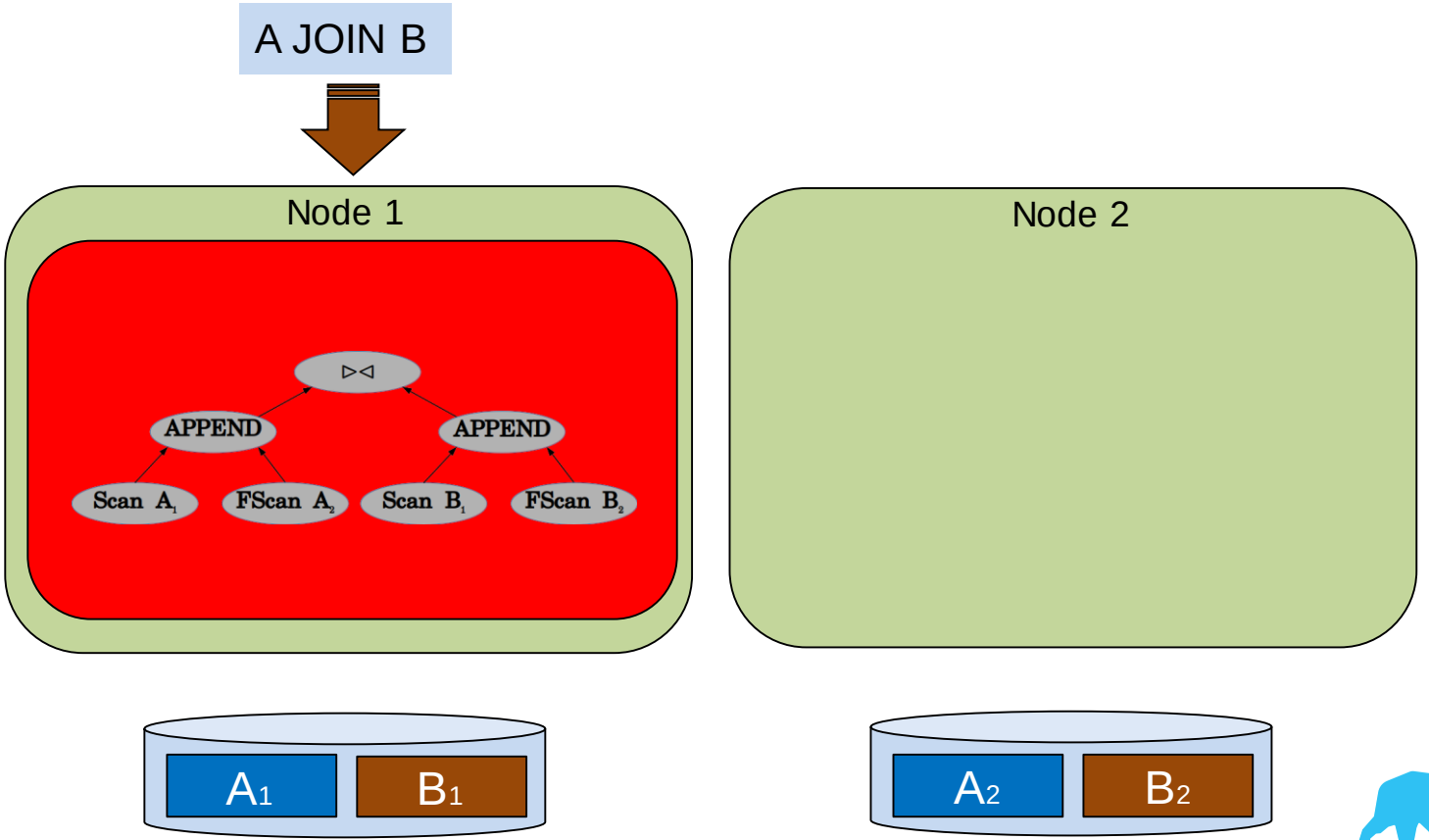
FDW + Partitioning



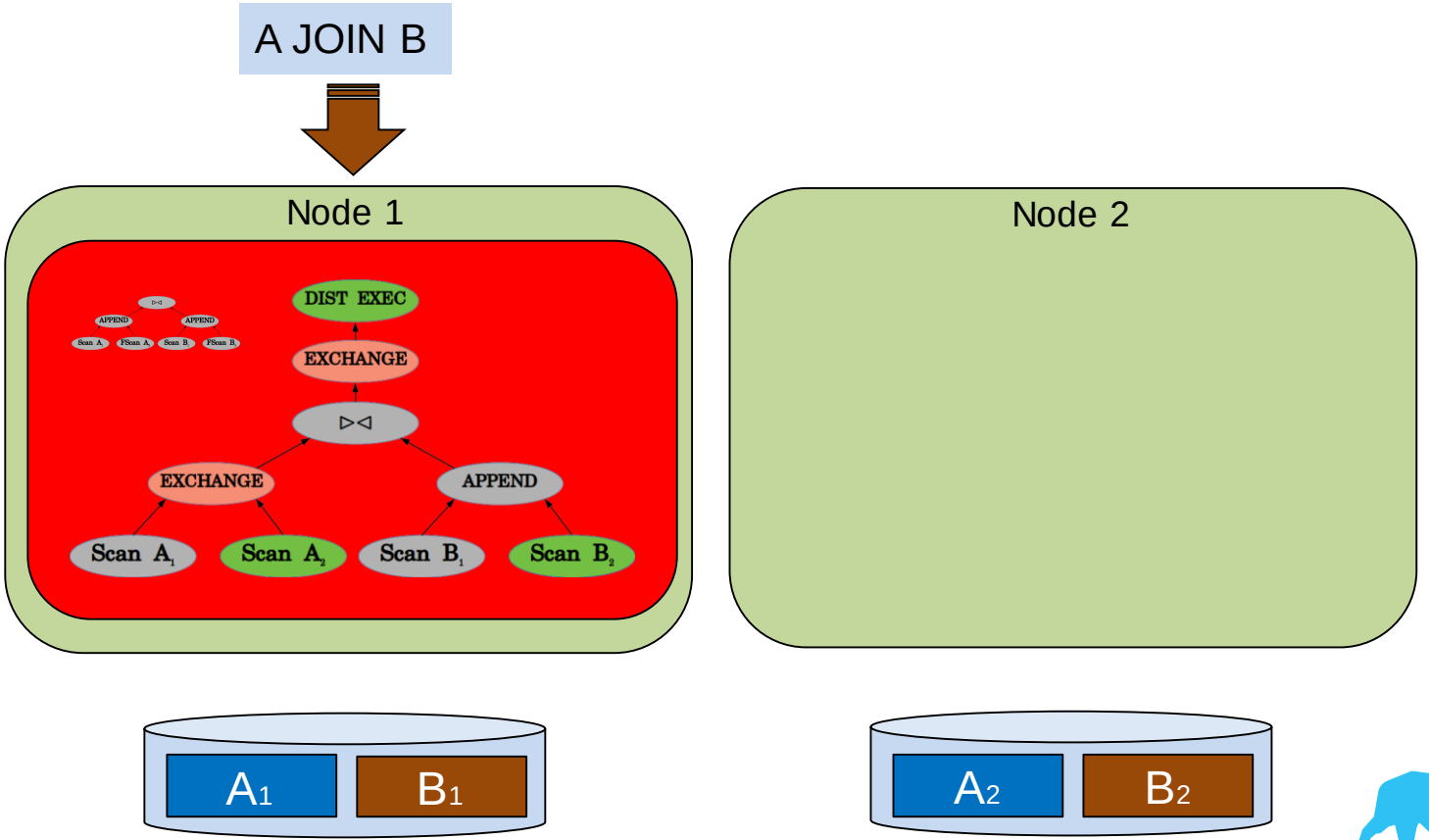
FDW + Partitioning

- Нет механизмов динамического решардирования
- Передает не план, а запрос
- Последовательный APPEND

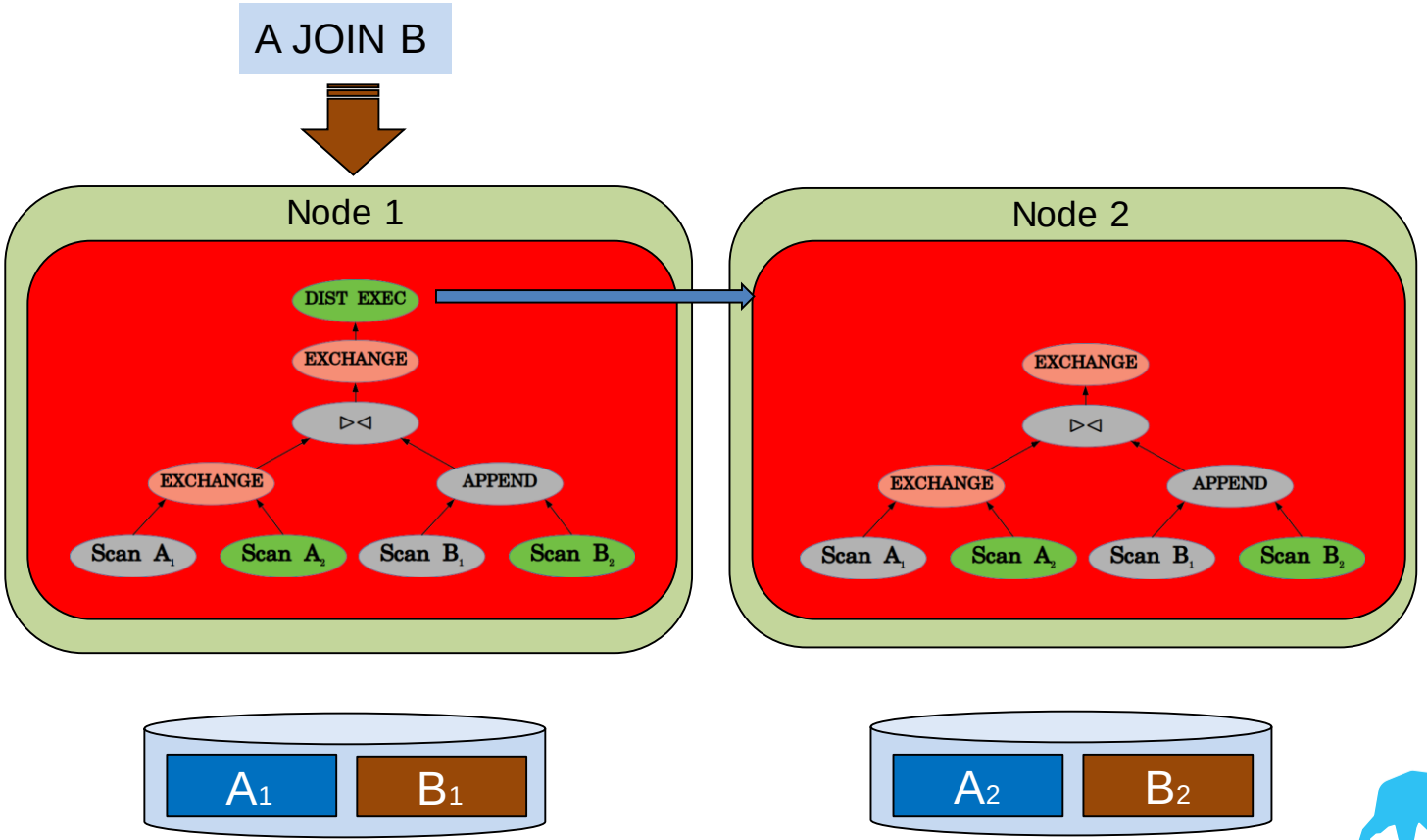
Шардман: выполнение запроса



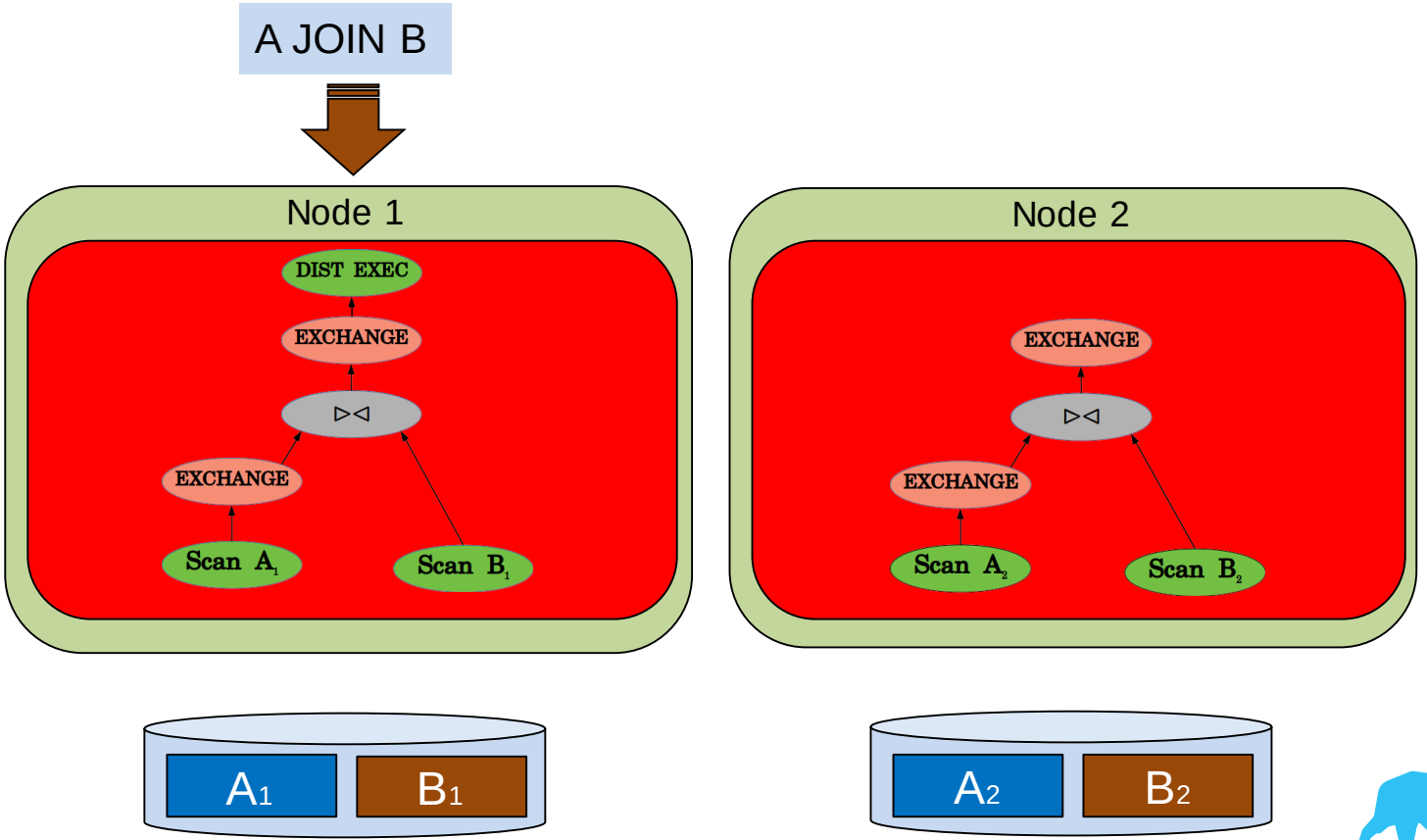
Шардман: выполнение запроса



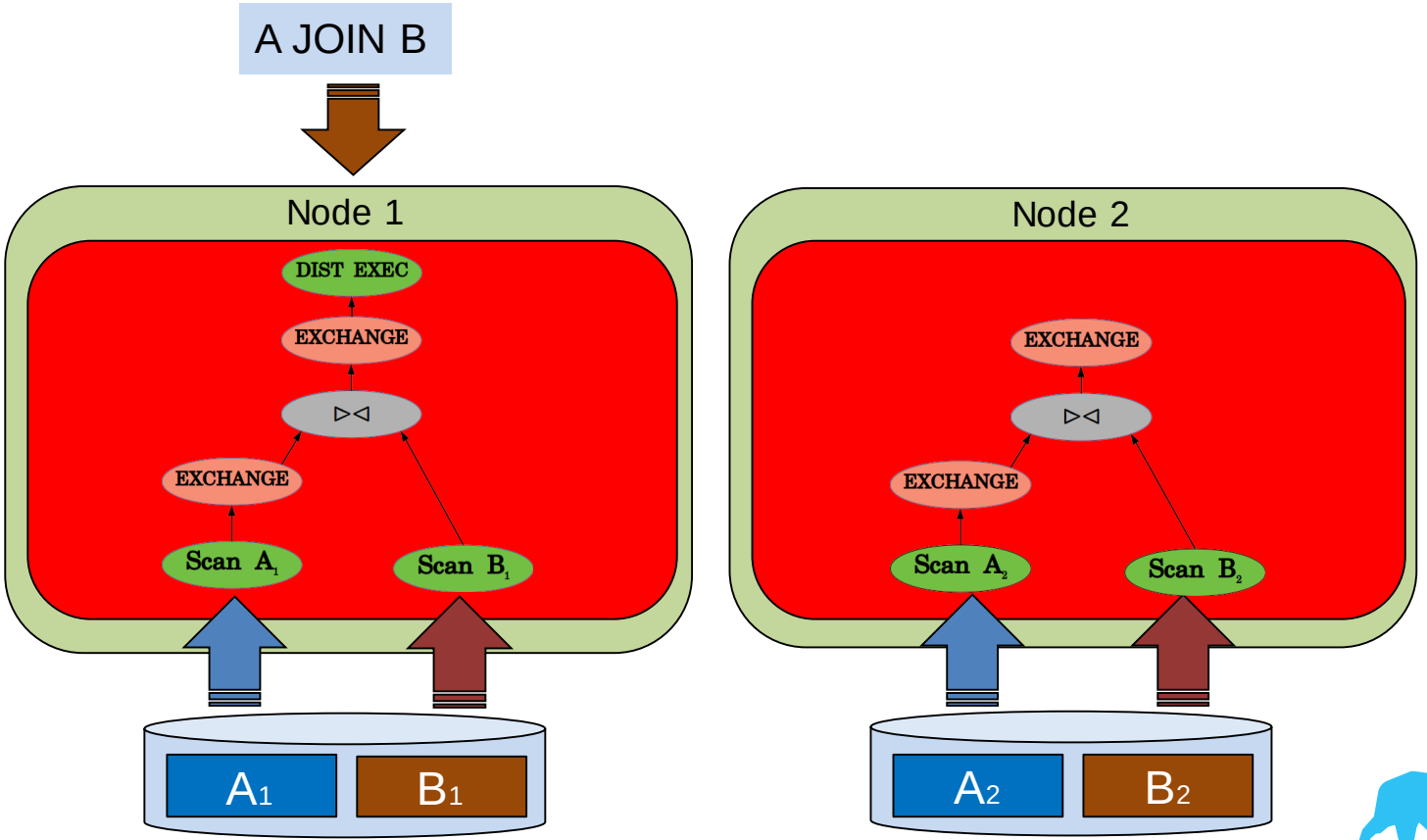
Шардман: выполнение запроса



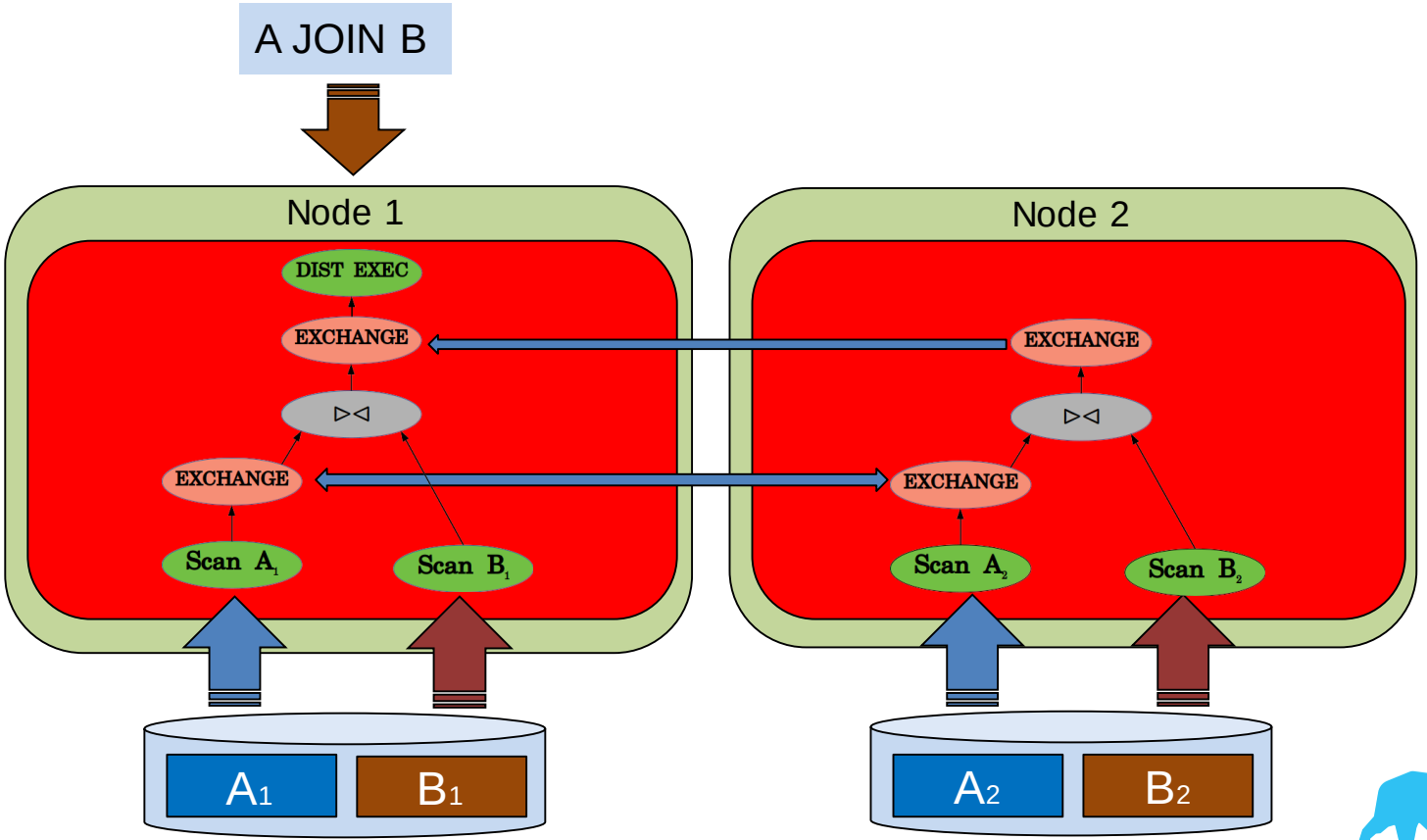
Шардман: выполнение запроса



Шардман: выполнение запроса

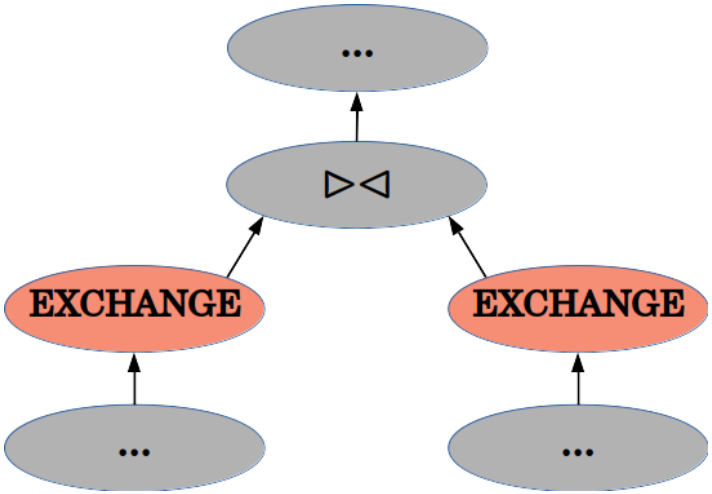


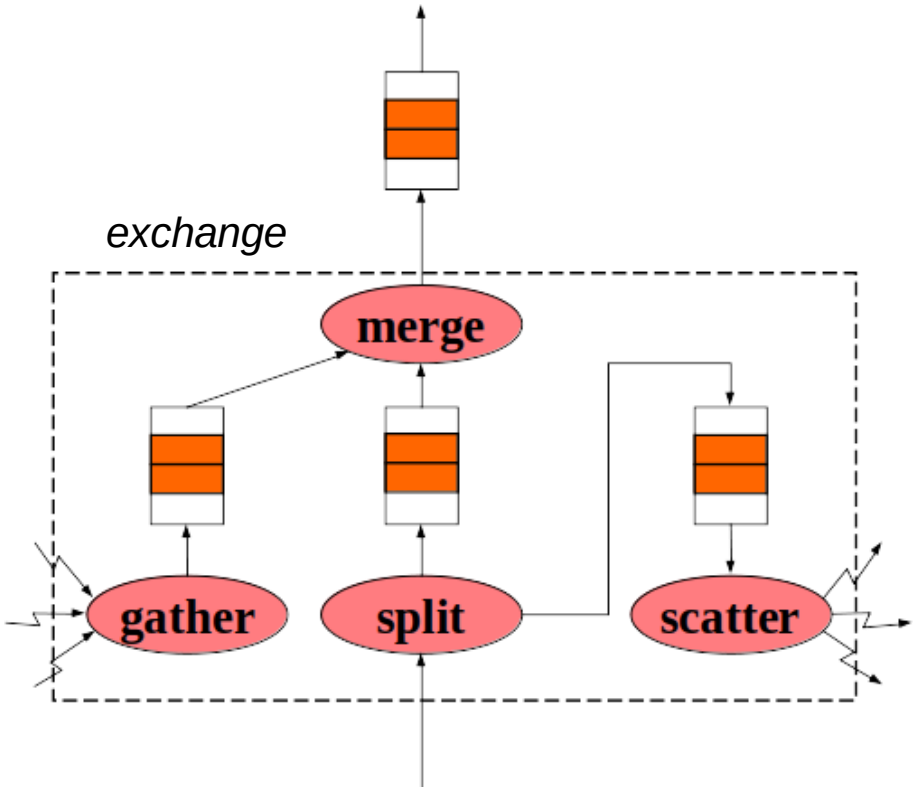
Шардман: выполнение запроса



Инкапсуляция параллелизма

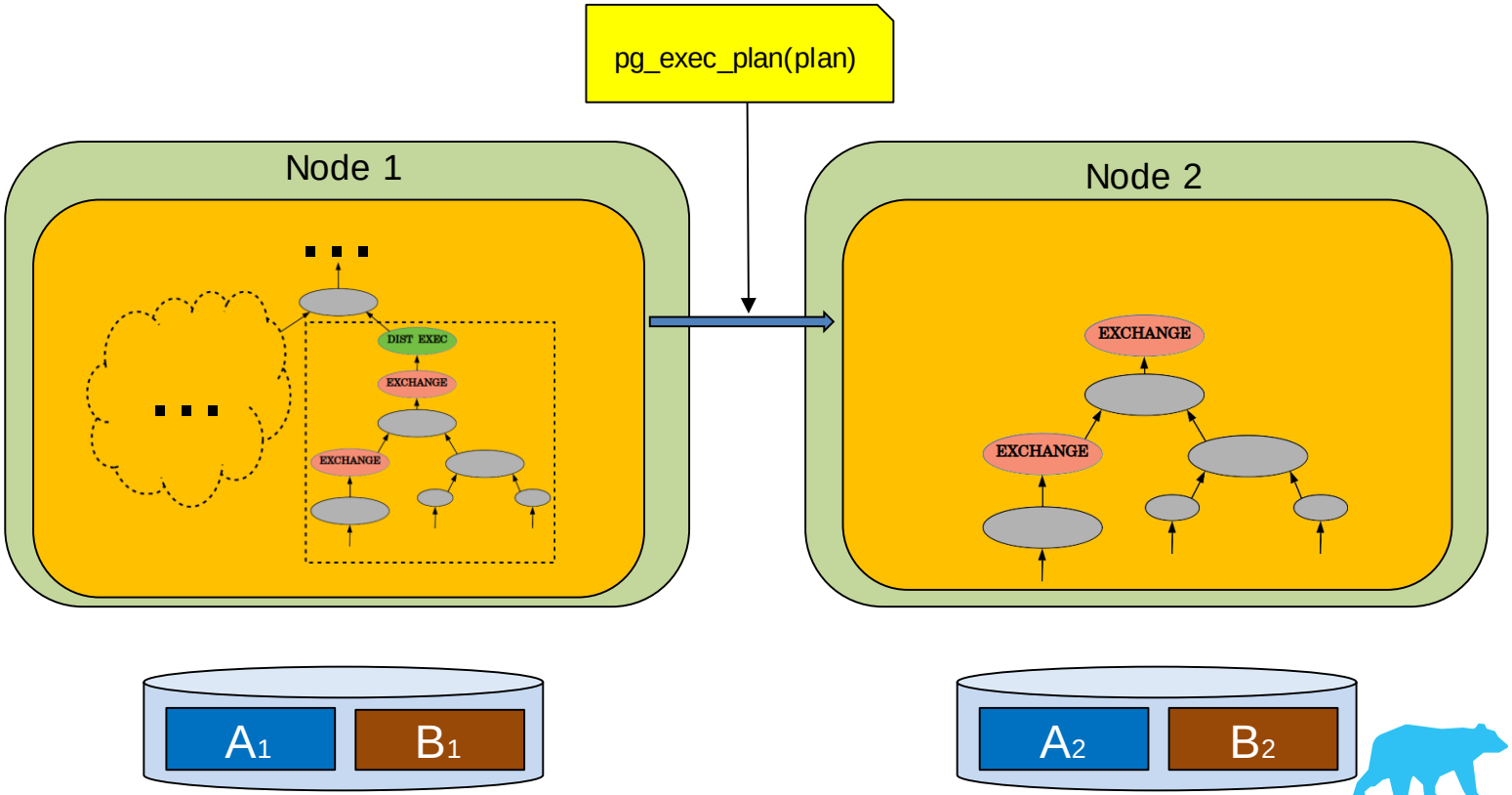
параллельное выполнение JOIN



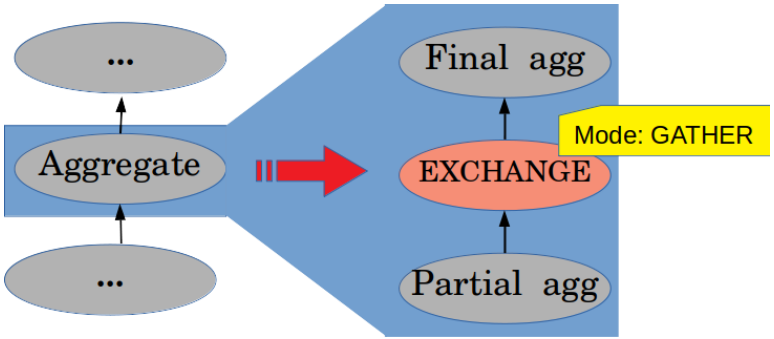


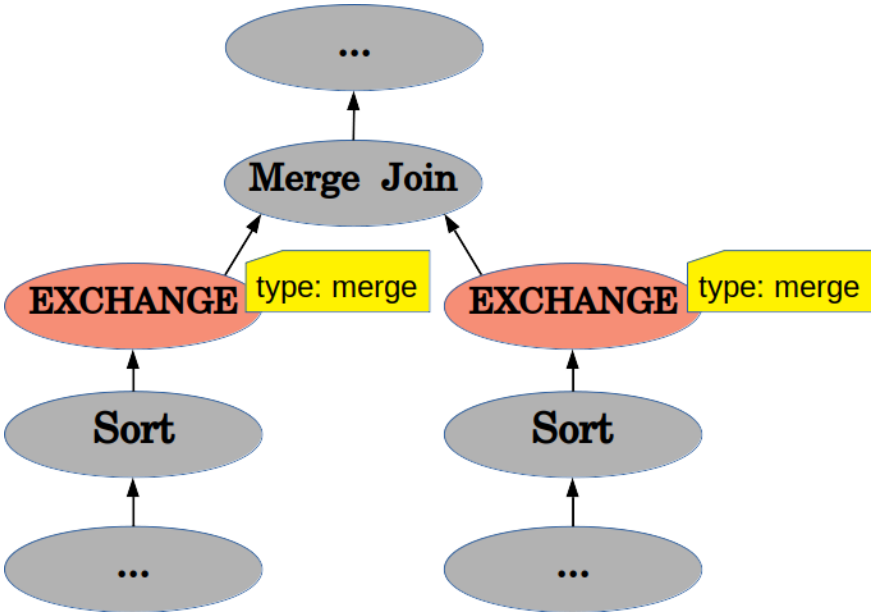
Рассылка планов

Dist Exec custom-node



To Do: агрегаты и сортировки

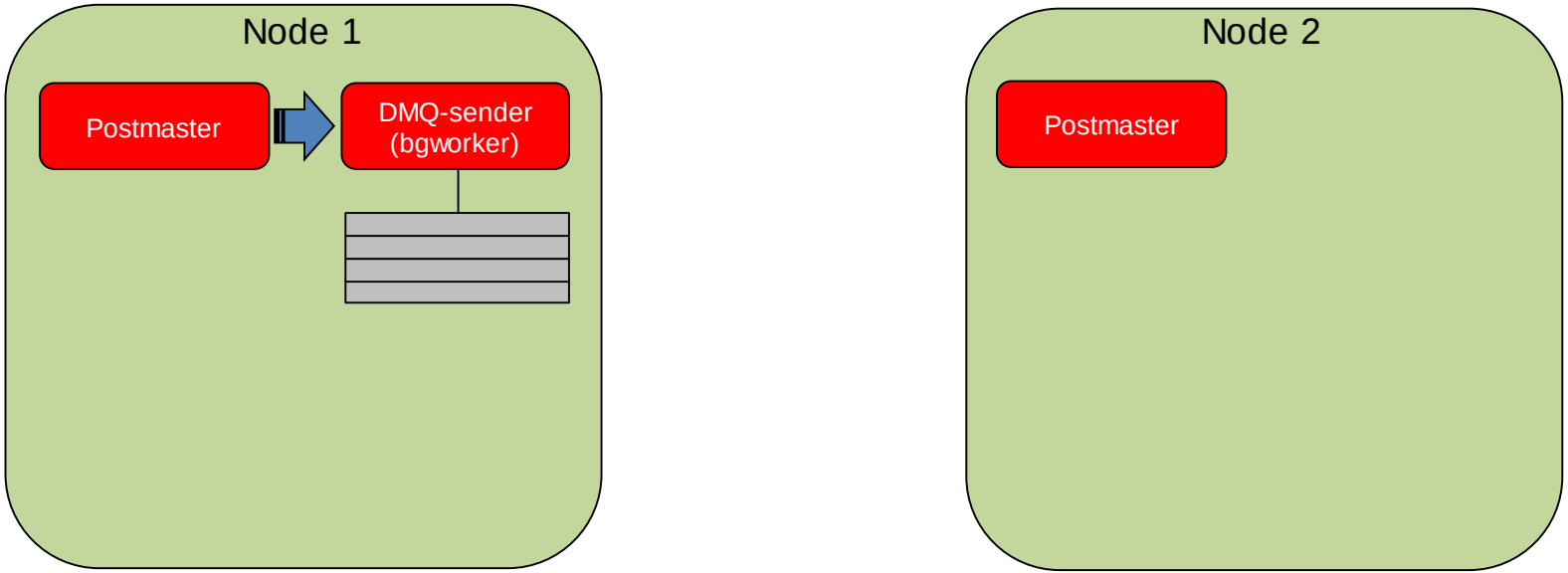




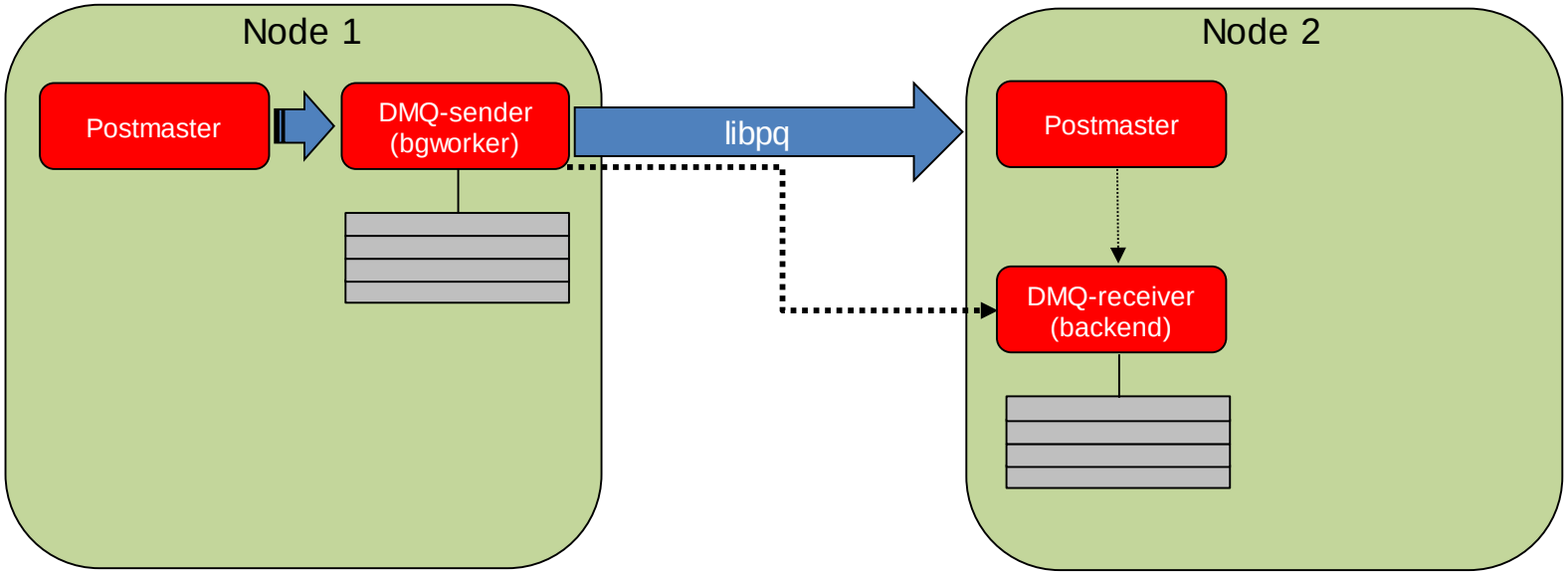
Message passing

- *Citus* – PostgreSQL receiver
- *Postgres-XL* - PostgreSQL receiver
- *Greenplum* – socket-based subsystem
- *Shardman* - DMQ

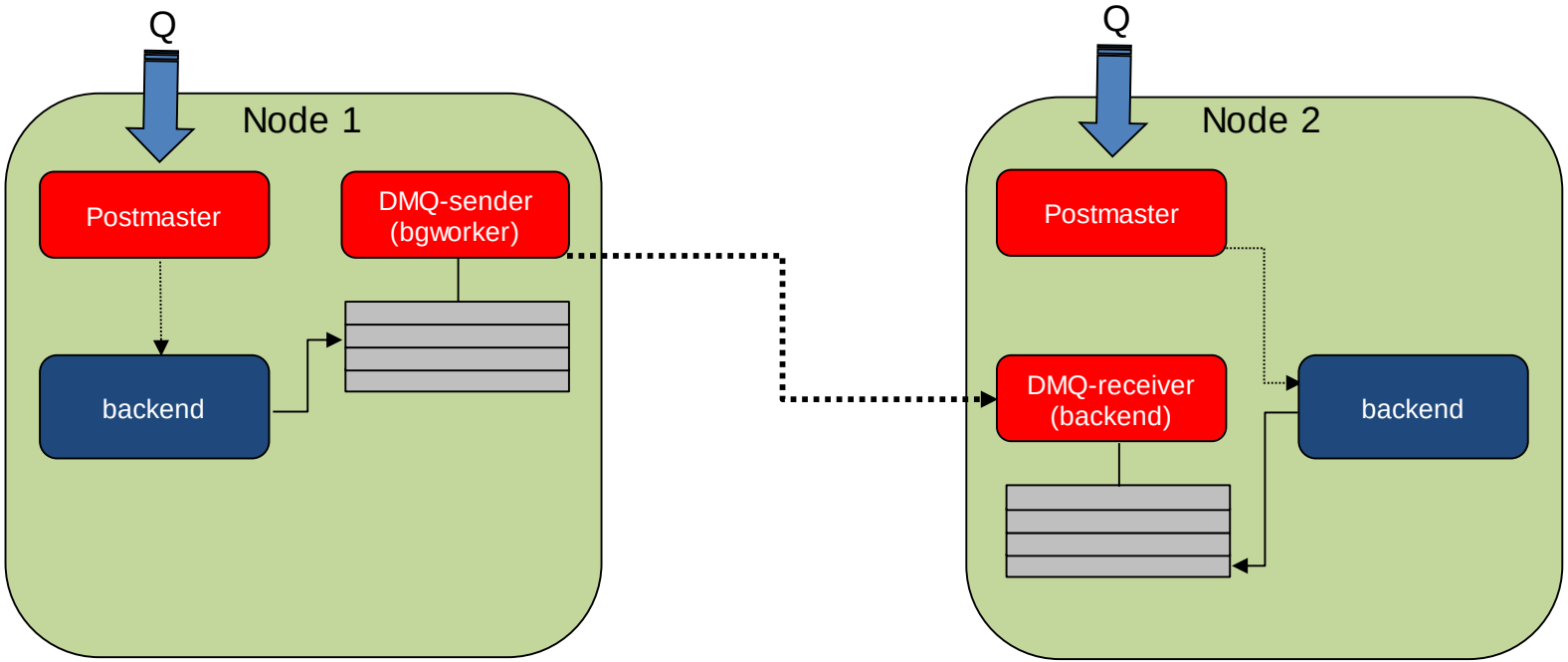
DMQ



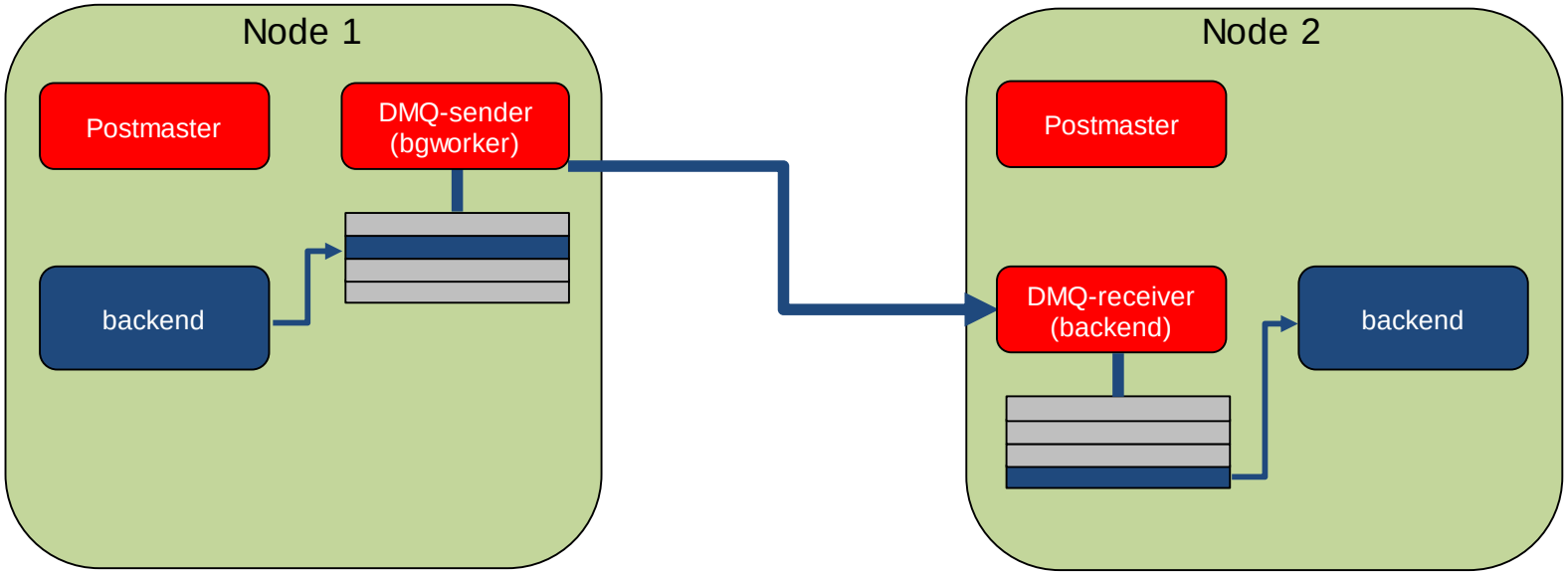
DMQ


Node 1
Postmaster
DMQ-sender
(bgworker)
libpq
Node 2
Postmaster
DMQ-receiver
(backend)

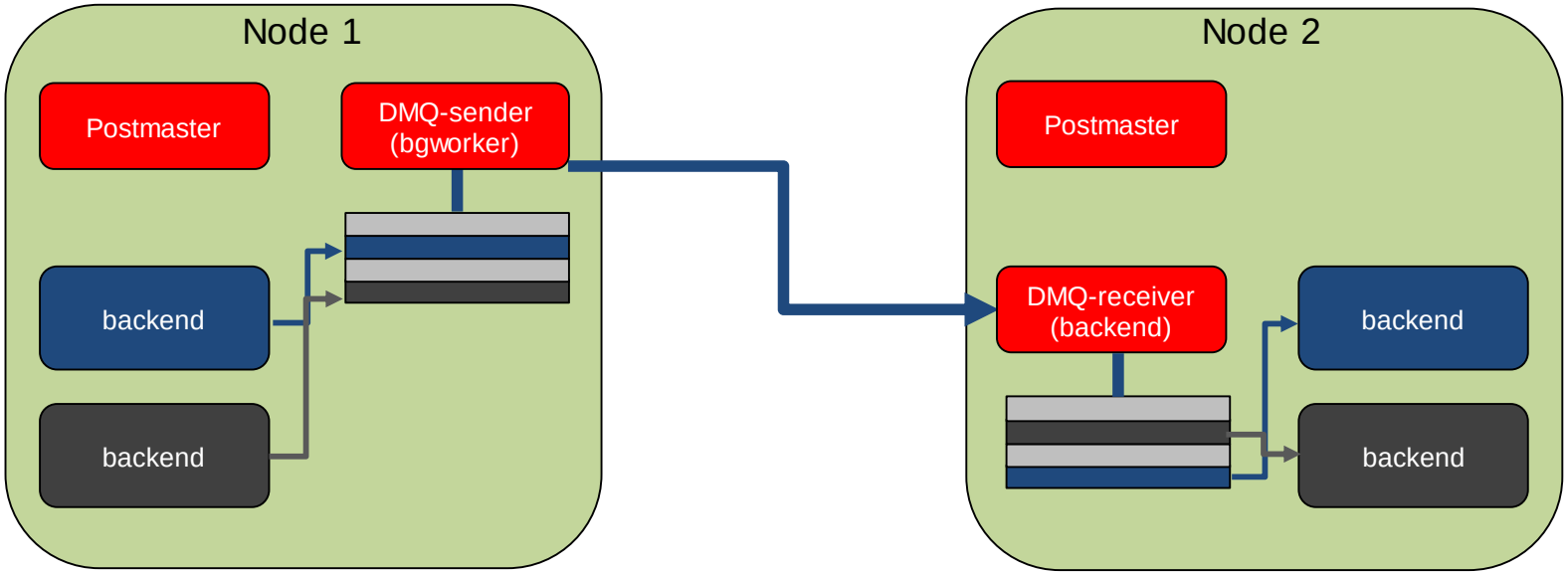
DMQ


Q
Node 1
Postmaster
DMQ-sender
(bgworker)
backend
Q
Node 2
Postmaster
DMQ-receiver
(backend)
backend

DMQ


Node 1
Postmaster
DMQ-sender
(bgworker)
backend
Node 2
Postmaster
DMQ-receiver
(backend)
backend

DMQ


Node 1
Postmaster
DMQ-sender
(bgworker)
backend
backend
Node 2
Postmaster
DMQ-receiver
(backend)
backend
backend

Что получилось

- Все PostgreSQL-инстансы равнозначны
- Только libpq соединения между инстансами
- Весь параллелизм инкапсулирован в custom-нодах планнера
- Параллелизм задействуется только для той части плана, где это необходимо
- Шардинг и прунинг выполняются средствами partitioning'а и FDW

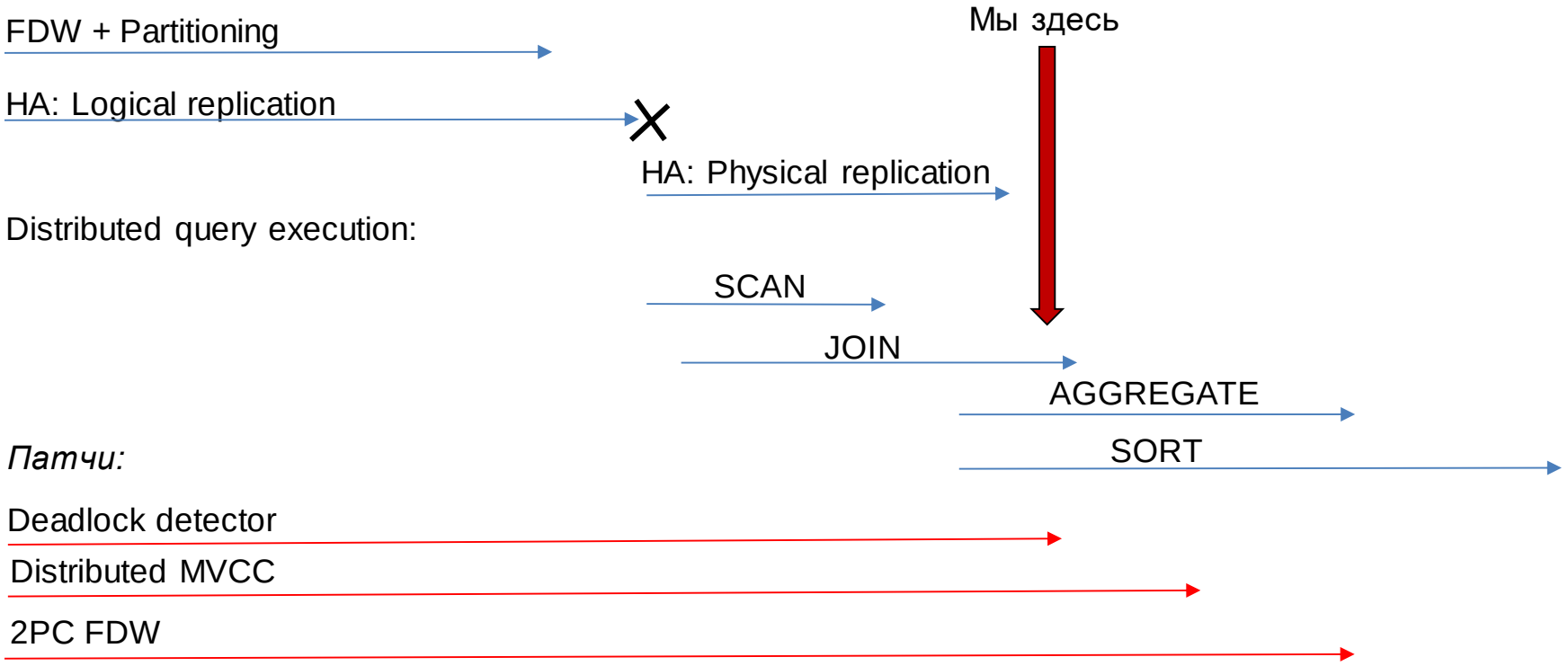
Шардман: OLTP

- Координатором может выступать любой узел
- План запроса создается однократно
- При использовании только локальных данных инстанса выполнение запроса не отличается от ванильного PostgreSQL
- Задействуется минимально необходимое количество инстансов, на которых выполняется минимально необходимый подплан

Шардман: OLAP

- Возможность параллельной загрузки данных
- SCAN, JOIN задействуют все инстансы (в перспективе агрегаты и сортировки)
- Доступны все планнеры постгреса

Roadmap



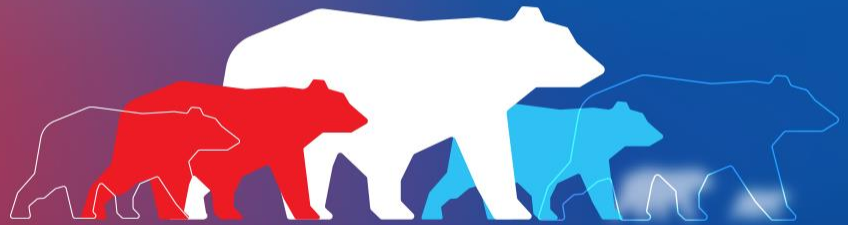
Ссылки

1. https://github.com/postgrespro/shardman/
2. https://github.com/postgrespro/postgres_cluster

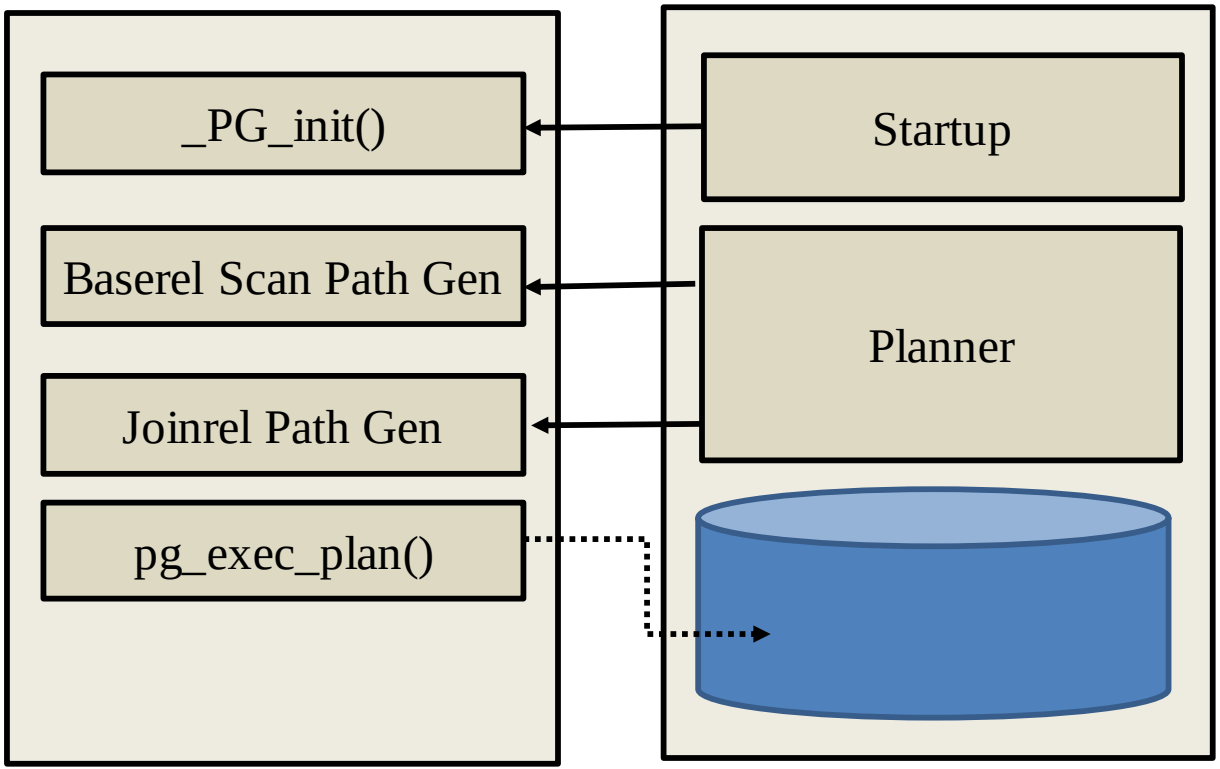
Вопросы?

HighLoad++ Siberia 2019

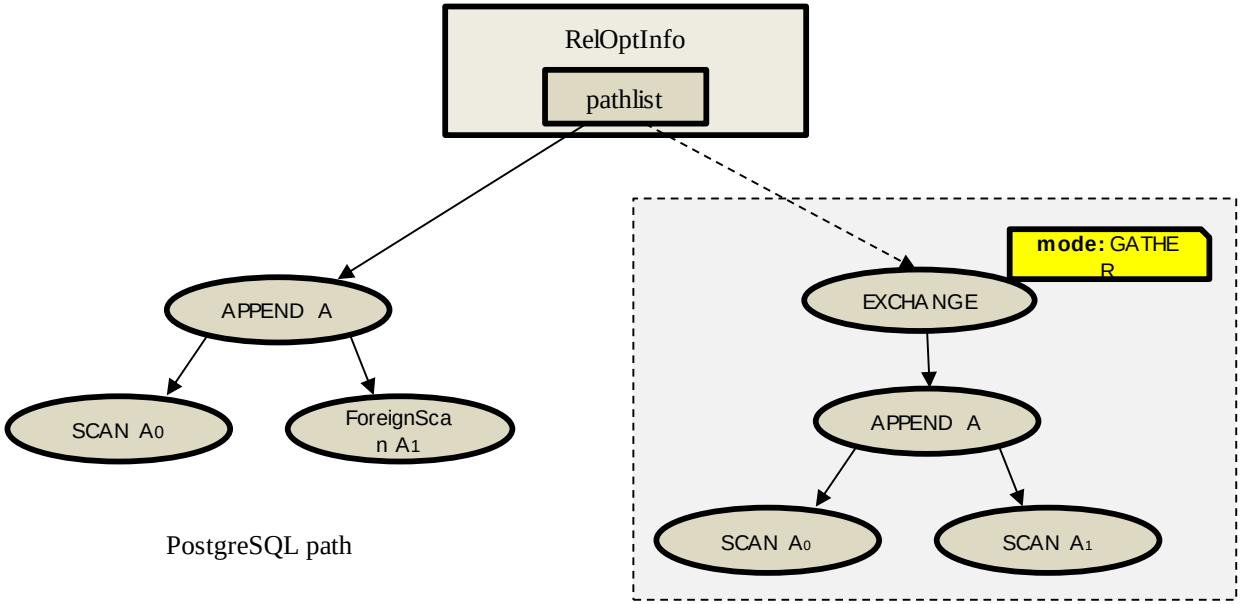
Профессиональная конференция для разработчиков высоконагруженных систем



Архитектура



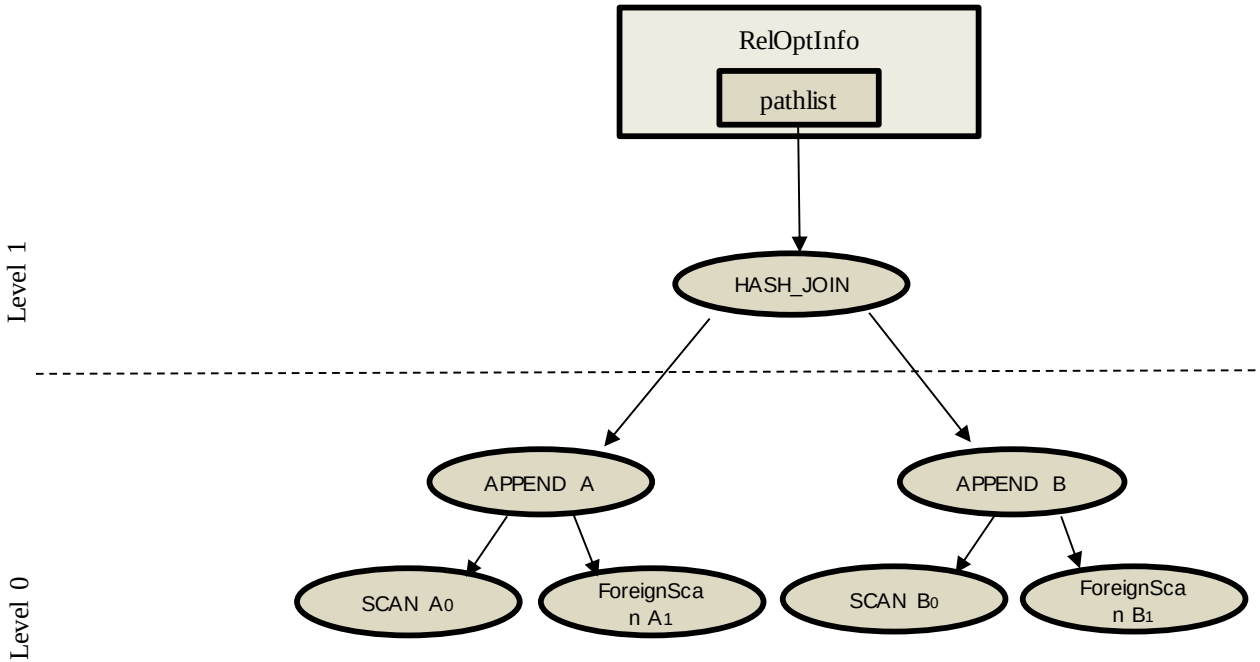
Baserel path generator



PostgreSQL path

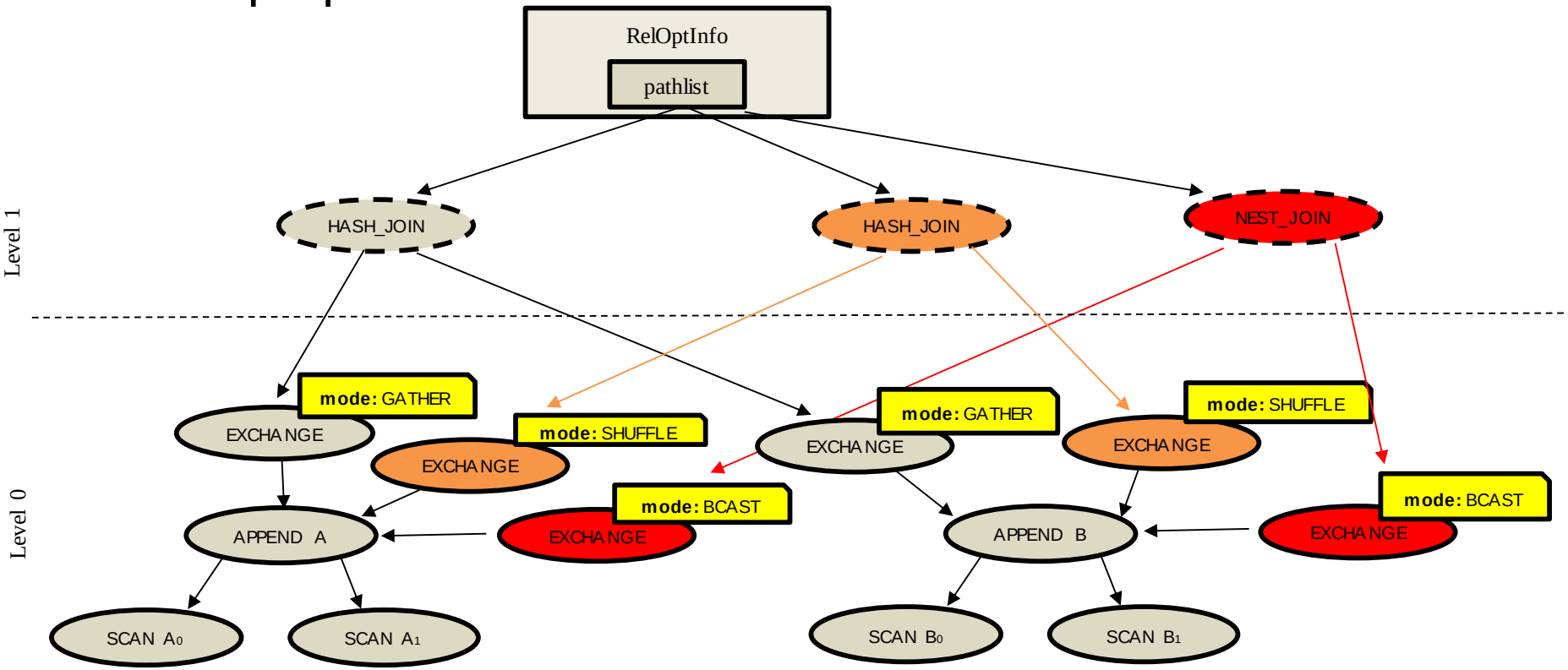
Shardman proposal

JOIN Path generator



JOIN Path generator

shardman proposal



Postgres-XL

SELECT * FROM A,B WHERE a.payload=b.id;