Beyond the pushdowns – distributed query planning and execution

Alexander Korotkov, Andrey Lepikhov

Postgres Professional

2019 October 16

Work author: Andrey Lepikhov

- Ph.D. in Parallel DBMS'es
- Core Developer in Postgres Professional
- Specialized in following PostgreSQL areas:
  - WAL,
  - Planner,
  - B-tree/GiST/SP-GiST access methods,
  - VACUUM.

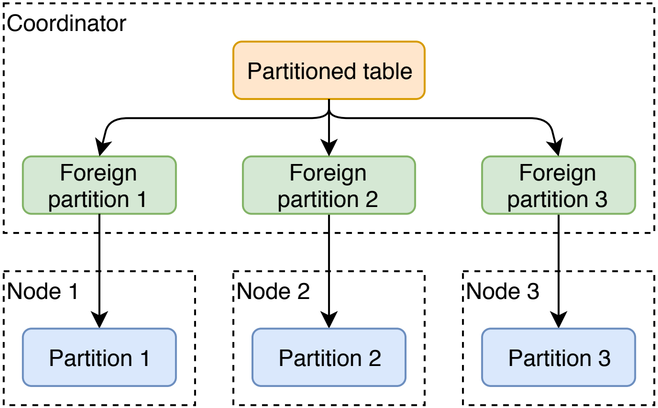
Speaker: Alexander Korotkov

- PostgreSQL Major Contributor & Committer,
- Contributed to indexing, SQL/JSON implementation, multicore optimizations, extensions and more,
- Chief Architect & Co-founder in Postgres Professional,
- Ph.D. in Computer Science,
- 3-times GSoC mentor.

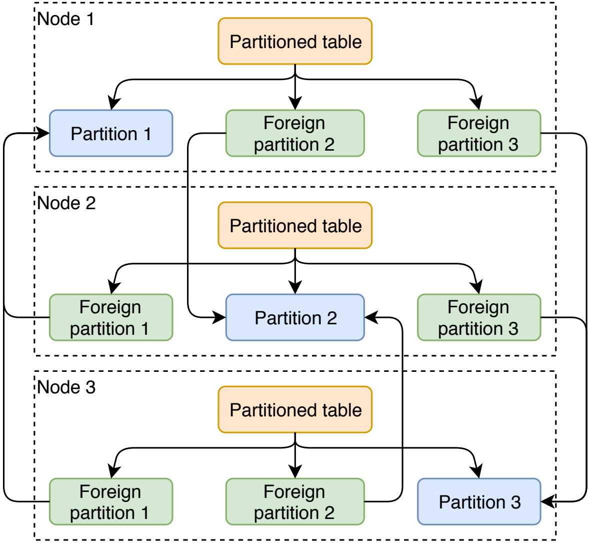
What is sharding in PostgreSQL?

Sharding = Partitioning + FDW

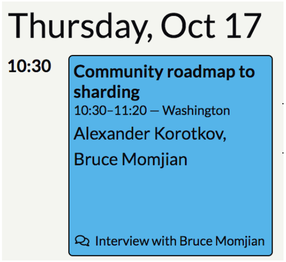
Sharding scheme 1



Sharding scheme 2



Do you need to go deeper?
Come visit this talk!



Thursday, Oct 17

10:30

Community roadmap to sharding

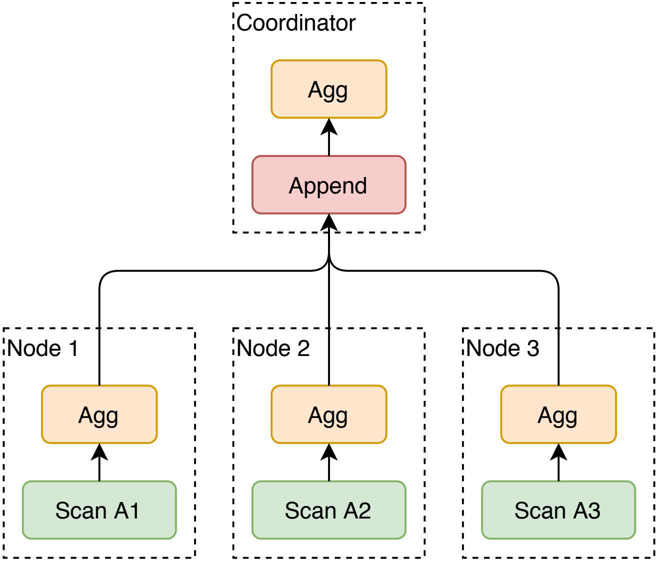
10:30–11:20 — Washington

Alexander Korotkov, Bruce Momjian

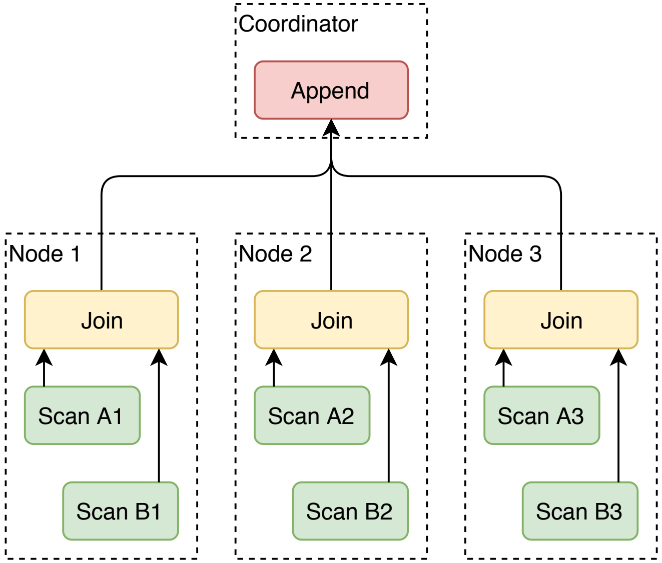
Interview with Bruce Momjian

What
Partitioning + FDW
can do?

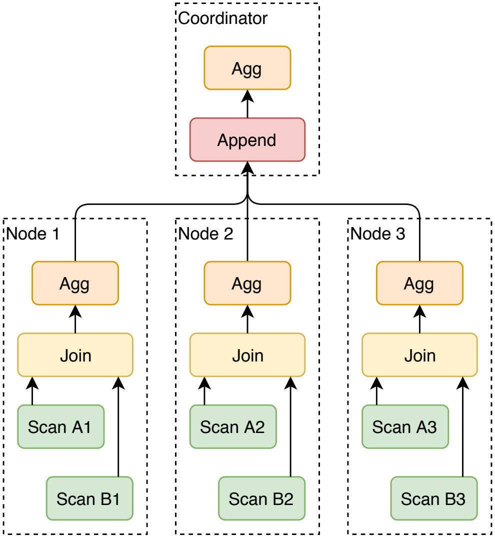
FDW can pushdown aggregates



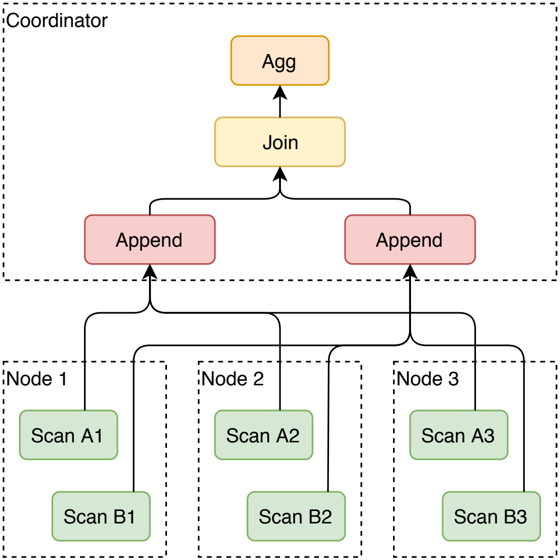
... and joins


Coordinator
Append
Node 1
Join
Scan A1
Scan B1
Node 2
Join
Scan A2
Scan B2
Node 3
Join
Scan A3
Scan B3

... and both

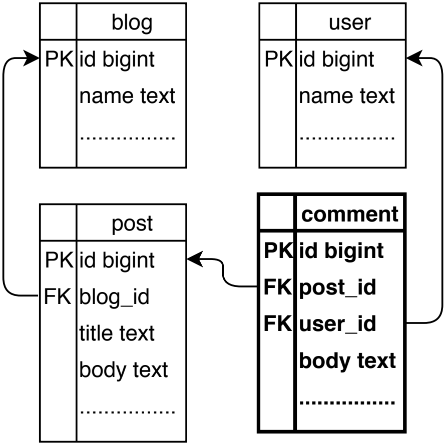

Coordinator
Agg
Append
Node 1
Agg
Join
Scan A1
Scan B1
Node 2
Agg
Join
Scan A2
Scan B2
Node 3
Agg
Join
Scan A3
Scan B3

If partitioning doesn’t match, then not so effective



Nice proposal ...

... but problematic to implement (1/2)



... but problematic to implement (2/2)

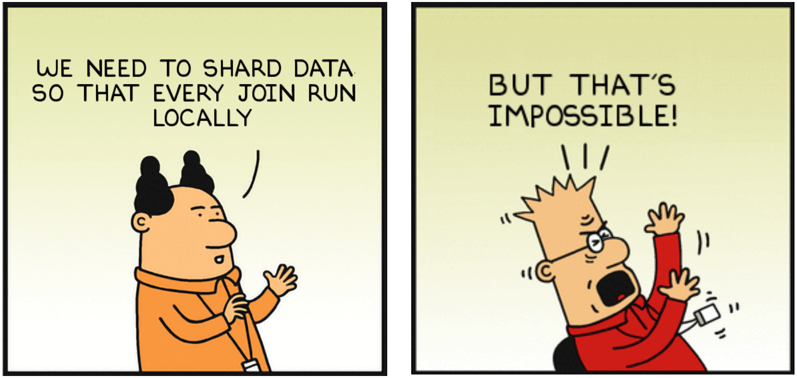
- Might need to shard on post_id.

```
SELECT p.category, count(*)
FROM comment c JOIN post p ON p.id = c.post_id
GROUP BY p.category;
```

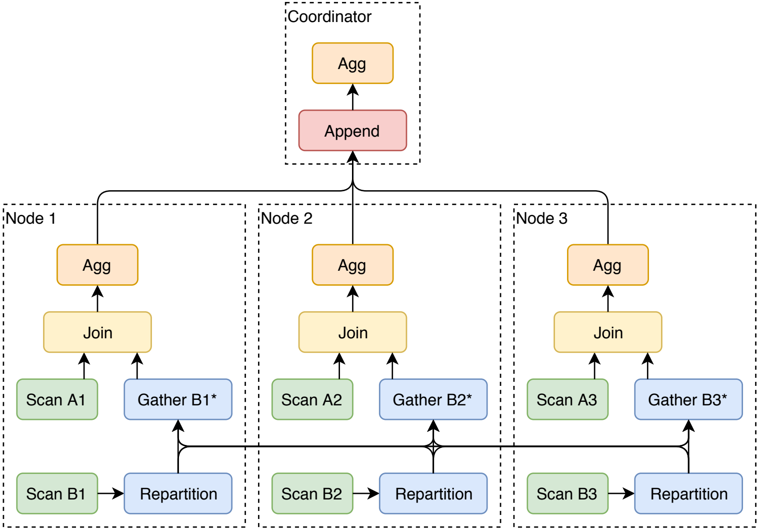
- Might need to shard on user_id.

```
SELECT u.source, count(*)
FROM comment c JOIN user u ON u.id = c.user_id
GROUP BY u.source;
```

... but problematic to implement



Repartition



Map-reduce?

Repartiton $\approx$ Map-reduce

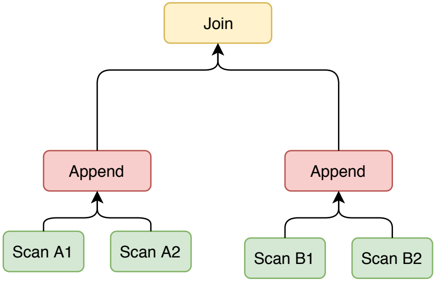
Map-reduce = SRF + Repartition + Aggregate

Shardman

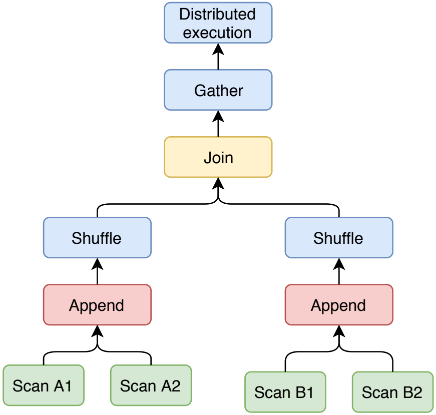
- `https://github.com/postgrespro/shardman`
- As EXTENSION as possible
- Automates sharding using partitioning + FDW
- Every instance is coordinator
- Configurable planning: FDW (best for OLTP) and distributed (best for OLAP)
- Hope to become pure extension

How does shardman plan/execute distributed (OLAP) queries?

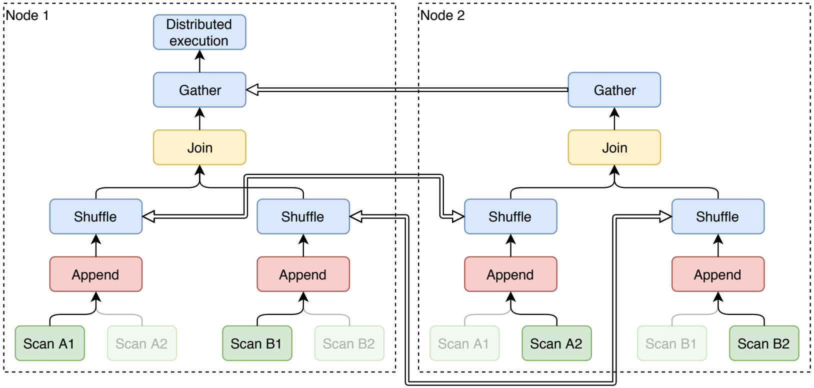
Distributed planning step 1: local plan



Distributed planning step 2: add distributed nodes



Distributed planning step 3: spread plans across the nodes



Stages of distributed query execution

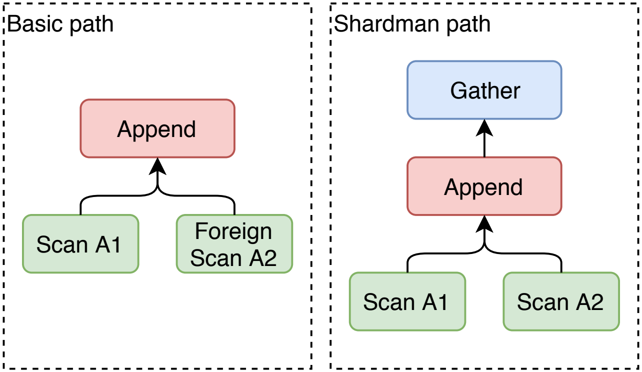
1. Prepare distributed query plan at coordinator node
2. Portable serialization of the plan, collect list of foreign servers
3. At the begin of query execution, pass the plan to each foreign server by FDW connection
4. Localize the plan - walk across scan nodes, remove unneeded scan nodes
5. Execute the plan

- Steps 1-3 for coordinator node
- Steps 3-4 for every involved node

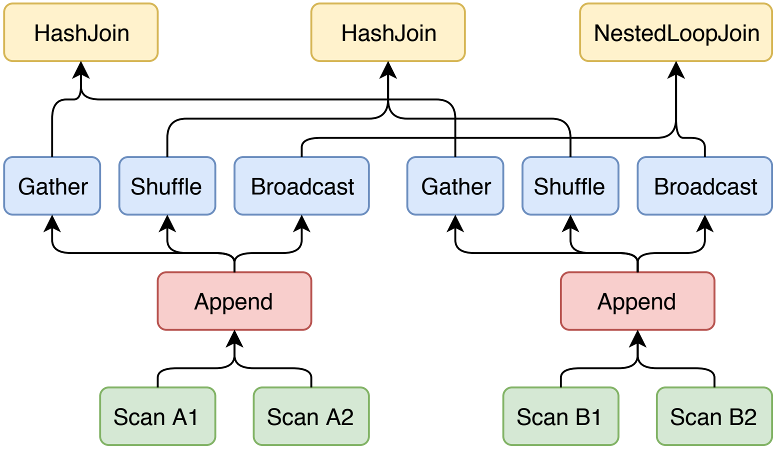
How distributed planning/execution integrates to PostgreSQL?

- Planner hooks: set_rel_pathlist_hook, set_join_pathlist_hook,
- Custom node: ExchangePlanNode,
- Portable plan serialization/deserialization. 1

set_rel_pathlist_hook



set_join_pathlist_hook



ExchangePlanNode

- Compute destination instance for each incoming tuple
- Transfer the tuple to the corresponding EXCHANGE node at the instance
- If destination is itself – transfer the tuple up by the plan tree
- Any distributed plan has EXCHANGE node in gather mode at the top of the plan: collect all results at the coordinator node.

Modes:

- **Shuffle** – transfer tuple corresponding to distribution function
- **Gather** – gather all tuples at one node
- **Broadcast** – transfer each tuple to each node (itself too)

Portable plan serialization/deserialization

- Patch nodeToString(), stringToNode() code.
- Serialization replaces OIDs with object names.
- Deserialization replaces object names back to OIDs.
- pg_exec_plan(plan text) deserializes, localizes and launches execution of the plan.

PostgreSQL core modifications

- Patch nodeToString(), stringToNode() code.
- Change partitioning code in the planner: partitioning of joinrel can be changing according to path (May be we transfer partitioning-related fields from RelOptInfo to Path structure?)

Status

- WIP
- Need to patch PostgreSQL core.
- HashJoin, NestedLoopJoin and HashAgg are implemented, MergeJoin and GroupAgg are in TODO list.
- Observed up to 5-times improvement in comparison with FDW on 4-nodes cluster (async execution!).
- `https://github.com/postgrespro/shardman` – go try it.

Thank you for attention!