Parallel Sequential Scan

- Robert Haas, Amit Kapila | PGCon 2015

Overall Status

- PostgreSQL 9.4 includes the basic facilities that will be used to implement parallel query (dynamic background workers, dynamic shared memory, shared memory message queues).
- PostgreSQL 9.5 includes most of the plumbing needed for parallel computation (error propagation, parallel mode/contexts).
- Working patches exist for parallel sequential scan, but were not committed to PostgreSQL 9.5 due to unresolved issues.

New in 9.5: Message Propagation

- Background workers can talk to user backends using the frontend-backend protocol.
- Protocol messages are sent via a shared memory message queue (shm_mq).
- In particular, if the background worker does something that generates an ERROR, WARNING, or other message, it can send that message to the master, and the master can receive it.

New in 9.5: Parallel Mode/Contexts

- Using parallel contexts, backend code can launch worker processes.
- Various pieces of state are synchronized from the parallel group leader to each worker (more on that in a minute).
- Neither the master nor the workers are permitted to make permanent modifications to any of the synchronized state while parallelism is active.
- No writes are allowed.
- Lots of backend code can run just fine in a parallel worker!

What Gets Synchronized?

- Libraries dynamically loaded by PostgreSQL.
- Authenticated user ID and current database.
- All GUC values.
- XID for current and top level transactions.
- XIDs that appear as committed.
- Combo CID mappings.
- Active and transaction snapshots.
- Current user ID and security context.

Patches for 9.6 (1 of 2)

- Heavyweight Lock Handling for Parallel Mode/Contexts
  - Must prevent unprincipled deadlocks between parallel workers.
- Assessing Parallel Safety
  - Every function in pg_proc is labelled to indicate whether it can be used in parallel mode. Most can!
  - Query planner is modified to search the query tree for unsafe functions, or any operation that writes data.

Patches for 9.6 (2 of 2)

- Parallel Seq Scan
  - General Executor Support For Parallelism
  - New Executor Nodes: Funnel, Partial Seq Scan
  - Might get split into several smaller patches

Parallel Seq Scan - New Nodes

- **Funnel**
  - Has one child, runs multiple copies in parallel.
  - Combines the results into a single tuple stream.
  - Can run the child itself if no workers available.
- **Partial Seq Scan**
  - Scans part of a relation sequentially.
  - Specifically, the part not scanned by any other copy of the same partial seq scan.

Parallel Seq Scan – Example Plan

- Funnel

  Number of Workers: 4

  -> Partial Seq Scan on tbl_parallel

- Each worker will scan part of the tbl_parallel; together, they will scan the whole thing.

Parallel Seq Scan – Information Sharing

- To perform parallel scan master and worker backend needs to share some information
  - Planned Stmt which needs to be executed by each worker
  - Bind Parameters
  - PARAMS_EXEC parameters (Execution time params required for evaluation of subselects)
  - Tuple Queues, to send tuples from worker to master backend
  - Instrumentation information required by Explain or other stats required by external utilities like pg_stat_statements

Parallel Seq Scan – Tuning Parameters

- parallel_degree - Maximum number of parallel workers that can be allocated to a particular parallel operation
- cpu_tuple_comm_cost - Cost of CPU time to pass a tuple from worker to master backend.
- parallel_setup_cost - Cost of setting up shared memory for parallelism, and launching workers.

Parallel Workers

- Parallel workers are launched at the start of funnel node execution
- Parallel workers will be stopped
  - As soon as last tuple is retrieved
  - During rescan
  - At end of execution
- Parallel workers will execute Partial Seq Scan node and produce tuples which are sent back to master backend

Parallel Workers – Work Allocation

- Two different strategies have been considered to allocate work for backend workers
  - Block-By-Block and Fixed Chunks
- Performance measurements didn't show much difference between the approaches, at least on the machines we tested.
- Preferred Block-By-Block, as that will allow work to be distributed dynamically based on the work finished by individual worker.

Performance Data

Common non-default settings

shared_buffers=8GB; min_wal_size=5GB; max_wal_size=10GB

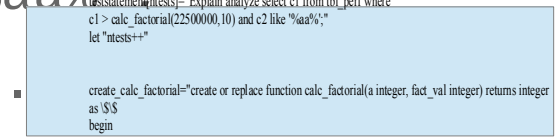
checkpoint_timeout =30min; max_connections=300;

max_worker_processes=100;

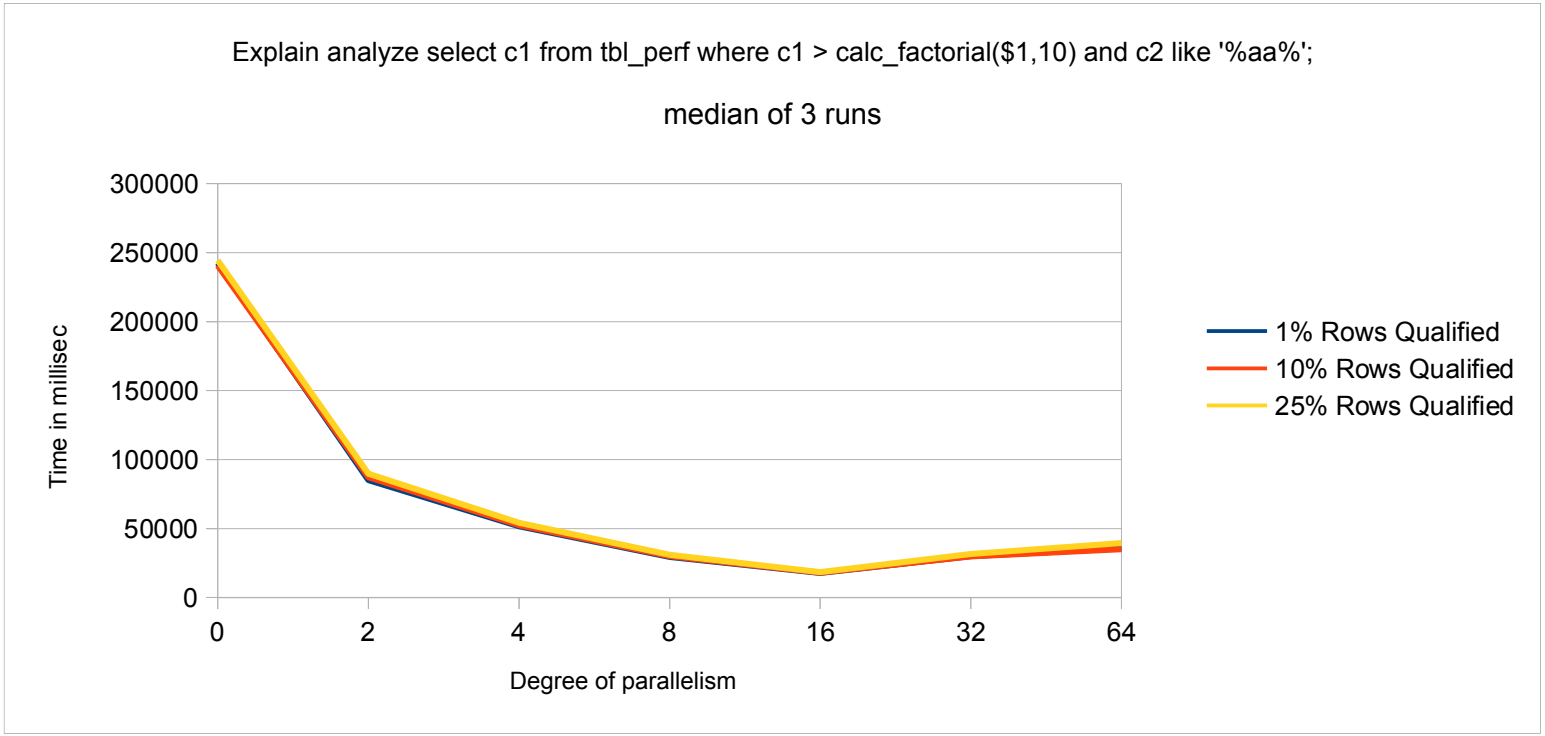
Test setup

```
create table tbl_perf(c1 int, c2 char(1000));

insert into tbl_perf
values(generate_series(1,30000000),'aaaaa');

Explain analyze select c1 from tbl_perf where c1 >
calc_factorial($1,10) and c2 like '%aa%';
```

Script used to take data is attached.



Performance Data



- With increase in degree of parallelism (more parallel workers), the time to complete the execution reduces.
- Along with workers, master backend also participates in execution due to which you can see more time reduction in some cases.
- After certain point, increasing number of workers won't help.

Future Work

Wondering why 2 new nodes (Funnel and PartialSeqScan) have been added?

Future Work – Join Pushdown (1 of 2)

- Example for Join Evaluation
  Nested Loop
  -> Seq Scan on foo
  -> Index Scan on bar
  Index Cond: bar.x = foo.x
- Now, if a parallel sequential scan is cheaper than a regular sequential scan, we can instead do this:
  Nested Loop
  -> Funnel
  -> Partial Seq Scan on foo
  -> Index Scan on bar
  Index Cond: bar.x = foo.x
- The problem with this is that the nested loop/index scan is happening entirely in the master.

Future Work – Join Pushdown (2 of 2)

- We can transform the plan to
  Funnel
  -> Nested Loop
  -> Partial Seq Scan on foo
  -> Index Scan on bar
  Index Cond: bar.x = foo.x
- The will allow the workers to execute the nested loop/index scan in parallel; we merge the results afterwards.

Future Work – Aggregate Pushdown

- We can push the Aggregates below the Funnel

```
HashAggregate
-> Funnel
    -> Partial Seq Scan on foo
        Filter: x = 1
```

- Assuming we have infrastructure to push the HashAggregates, we can convert it to

```
HashAggregateFinish
-> Funnel
    -> HashAggregatePartial
        -> Partial Seq Scan on foo
            Filter: x = 1
```

Thanks.

- Any questions?