Who Am I

Database Architect at DRW
Local Organizer of Chicago PostgreSQL User Group

PG Day Chicago is on April 26, 2024!
I never presented any talk about tuning parameters - why now?!
Tuning your database – what does it mean?

I thought I knew...

Until I started working for EDB!
And then it turned out that *tuning* means tuning parameters!
Why I never did it before?
Why people believe in the magic of parameters?
My goal today is to show
Why it almost does not matter
Why it is difficult to show?

- Tuning individual queries performance vs improving throughput
- It’s difficult to model a real-life workload
- It’s difficult to model a real-life concurrency
Tuning parameters can improve performance 10%, 20%, in some cases 50%.

Tuning queries can improve performance several (tens) times.

Tuning application can improve performance up to hundreds of time!
SELECT f.flight_no,
    f.actual_departure,
    count(passenger_id) passengers
FROM flight f
JOIN booking_leg bl ON bl.flight_id = f.flight_id
JOIN passenger p ON p.booking_id=bl.booking_id
WHERE f.departure_airport = 'JFK'
AND f.arrival_airport = 'ORD'
AND f.actual_departure BETWEEN
    '2023-08-08' and '2023-08-12'
GROUP BY f.flight_id, f.actual_departure;
Execution plan with default memory allocation

shared_buffers=128MB
work_mem=4MB
max_parallel_workers_per_gather=0

Execution time: 2.4 s
Execution plan with default memory allocation

shared_buffers=128MB  
work_mem=4MB  
max_parallel_workers_per_gather=2

Execution time 2.1 s
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Cost</th>
<th>Rows</th>
<th>Width</th>
<th>Actual Time</th>
<th>Rows</th>
<th>Loops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gather Merge</td>
<td>276065.38..276069.53</td>
<td>8</td>
<td>24</td>
<td>1726.858..1727.739</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Workers Planned: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Workers Launched: 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Buffers: shared hit=4546 read=171289</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Partial GroupAggregate</td>
<td>275065.36..275068.59</td>
<td>4</td>
<td>24</td>
<td>1722.810..1722.816</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Group Key: f.flight_id</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Buffers: shared hit=4546 read=171289</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Sort</td>
<td>275065.36..275066.42</td>
<td>425</td>
<td>20</td>
<td>1722.803..1722.806</td>
<td>54</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Sort Key: f.flight_id</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Increasing `work_mem`

- `shared_buffers=128MB`
- `max_parallel_workers_per_gather=2`
- `work_mem = 500MB`
  - 1.8 sec
- `work_mem = 1GB`
  - 1.8 sec
Increasing shared buffers (requires restart)

`shared_buffers=1GB`
`Work_mem=4MB/100MB/500MB`

1s

`shared_buffers=2GB`
`Work_mem=500MB`

1.1s
Let’s try something different!
Let’s take a closer look at the execution plans we have so far

Heap scan when looking for the departure dates between August 8 and 12.
Build the index

CREATE INDEX flight_actual_departure
    ON postgres_air.flight
    (actual_departure);

Execution time: 0.7 s
<table>
<thead>
<tr>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>→ Hash Join (cost=2243.76..267385.35 rows=473 width=20) (actual time=490.297..2186.655 rows=54 loops=3)</td>
</tr>
<tr>
<td>Hash Cond: (p.booking_id = bl.booking_id)</td>
</tr>
<tr>
<td>Buffers: shared hit=498 read=171481</td>
</tr>
<tr>
<td>→ Parallel Seq Scan on passenger p (cost=0.00..239646.72 rows=6797372 width=8) (actual time=0.521..1941.157 rows=5437898 loops=3)</td>
</tr>
<tr>
<td>Buffers: shared hit=192 read=171481</td>
</tr>
<tr>
<td>→ Hash (cost=2242.12..2242.12 rows=131 width=20) (actual time=4.722..4.725 rows=69 loops=3)</td>
</tr>
<tr>
<td>Buffers: shared hit=284</td>
</tr>
<tr>
<td>Buckets: 1024 Batches: 1 Memory Usage: 12kB</td>
</tr>
</tbody>
</table>
Build another index!

CREATE INDEX passenger_booking_id
ON postgres_air.passenger (booking_id);

Execution time: 60 ms
What will happen if we return parameters back to default?
The execution plan will remain the same (and the execution speed as well!)
Understanding the role of parameters

Communicating the hardware characteristics to PostgreSQL

Examples:
- RAM 16 GB/ shared_buffers 128MB
- RAM 16 GB/shared_buffers 4 GB/ work_mem 200MB/max_connections 1000
- random_page_cost 4
Application changes

Not necessarily NORM!

Examples:
• Using ‘=’ instead of ‘~’
• Column transform:
  \text{trunc(created\_dt)}=\text{CURRENT\_DATE}
• Committing each record
• Not committing until the batch end