random_page_cost

Why the default is 4.0 and should you lower it?

Tomas Vondra | Microsoft

<tomas@vondra.me> | https://vondra.me

POSETTE 2026

Agenda

- Purpose of random_page_cost
  - Came with the cost model in ~2000.
- How was the default 4.0 selected?
  - How was it determined / measured?
  - Can we repeat the process on current hardware?
- Is 4.0 still the right default?
  - Maybe we should lower it on SSD storage?
- Other effects
  - Indirect effects and limitations of the cost model.

Purpose of random_page_cost

Microsoft

- cost model
  - plan => sum of resources needed to execute the plan
  - assumption: more resources => takes longer
  - resources = CPU, disk I/O, ...
- cost parameters
  - seq_page_cost = 1.0
  - random_page_cost = 4.0
  - cpu_tuple_cost = 0.01
  - cpu_index_tuple_cost = 0.005
  - cpu_operator_cost = 0.0025
  - ...

I/O costs (usually) dominate

Microsoft

- disk I/O dominates the costs
  - combination of sequential I/O + random I/O
  - sequential I/O much faster than random I/O (even on SSDs)
- sequential scan

```
seq_page_cost * npages + cpu_tuple_cost * ntuples + cpu_operator_cost * ntuples
      1.0                     0.01                          0.0025
```

- index scan

```
random_page_cost * npages_accessed + cpu_index_tuple_cost * ntuples + ...
      4.0                                    0.005
```

Where does 4.0 come from?

Microsoft

- cost model introduced in ~2000
  - message by Tom Lane describing the approach
    https://www.postgresql.org/message-id/flat/14601.949786166@sss.pgh.pa.us
- hardware changed a lot
  - used to be spinning drives, now most storage is flash-based / SAN / ...
- Postgres was improved quite a bit
  - better efficiency of various code paths (originally dominated by I/O costs)
- the kernel improved a lot too
  - Linux read-ahead came later (in ~2004)

Maybe time to update the default?

Microsoft

- attempt to reproduce an experiment from 2000
  - the original script seems to be lost, based on recollections
- benchmark / measuring steps
  1. generate a table with random data (uniform distribution)
  2. read it sequentially and randomly
  3. calculate "per page" timings
  4. random_timing / sequential_timing => random_page_cost
- account for caching effects (index scans)
  - large data + direct I/O => caching effects do not matter

EXPLAIN ANALYZE

```
EXPLAIN ANALYZE SELECT * FROM t ORDER BY a;
                                QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using t_a_idx on t  (cost=0.42..206456.42 rows=1000000 width=8)
                   (actual time=12.904..109359.817 rows=1000000.00 loops=1)
   Index Searches: 1
   Buffers: shared hit=341218 read=661163 written=7271
 Planning Time: 0.453 ms
 Execution Time: 109581.587 ms

random_page_time = (109581.587 / 661163) = 0.166;
seq_page_time = (278.015 / 45455) = 0.006;

random_page_cost = (0.166 / 0.006) = 27.7;
```

Results

Microsoft

host	storage type	estimated random_page_cost
home	7.2k SATA (2x, RAID0)	140
home	SSD (4x SATA, RAID0)	15
home	SSD (4x NVMe, RAID0)	25
home	SSD (NVMe)	35
cloud	SSD (2x NVME, RAID0)	40
cloud	SAN (SSD, 80k IOPS, 1GB/s)	25

What do the results say?

Microsoft

- 4.0 never matched the "raw" random cost
  - at least judging by the results from SATA drives
- there is a difference between SSDs and spinning drives
  - SSDs do better, but random I/O still not 1:1 with sequential I/O (also much better sequential bandwidth)
- should you lower random_page_cost on SSDs?
  - SSDs certainly do better than spinning drives (25 < 140).
  - But don't the results suggest increasing the default?

Increasing random_page_cost?

Microsoft

- Is that a good idea?
  - try increasing random_page_cost (say to 30)
  - it will probably "fix" individual queries (in isolation)
  - likely to perform rather poorly for the whole system
- Why?
  - indirect caching effects
  - cost model not accounting for these effects

cost model

Microsoft

- inherently approximate / simplified
  - the whole point of having a model
- approximation
  - optimizer statistics approximate the data set (estimates)
  - cost calculations make various assumptions (independence, ...)
- simplification
  - a lot of stuff is opaque for Postgres (up to the operating system)
  - which part of data needed by query is currently in page cache?
  - no concept of an "active set"
  - limited consideration of concurrent activity (isolated queries)

memory as costed resource

Microsoft

- memory is not a costed resource
  - there's work_mem, but that's a limit for some operations
  - does not assign cost to "pushing data from page cache"
- consider reading 1GB from a 100GB table
  - a seqscan (100GB) may be faster than an index scan (1GB)
  - seqscan will "evict" a lot more data from the page cache
  - less of an issue for shared buffers (circular buffer)
- random I/O is more "localized"
  - only reads data matching query predicates
  - pushes less data from page cache

conclusions

Microsoft

- So should you lower random_page_cost?
  - maybe, something like 2.0 seems reasonable on SSDs
  - but monitor performance of the whole system
  - mitigated by: bitmap scans, narrow range selectivities
- not a question of "raw" I/O cost
  - tradeoff between local vs. global optimum (query vs. whole instance)
  - simple experiments can't give you a "correct" number
  - impossible to tune this on a query in isolation (global feature / caching)
  - has to be driven by monitoring of the system / evaluation of performance

cost model improvements

Microsoft

- there'll always be something "missing"
  - model = simplification / approximation of reality
  - largel model = requires more stats = analyze / planning costs
- model could consider a lot more stuff
  - costing of "memory" as a resource
  - better idea of what's in the cache / active set
  - AIO / prefetching
  - forward vs. backward scans (read-ahead only for forward scans)
  - ...

Other

Microsoft

- office hours
  https://vondra.me/office-hours/
- The real cost of random I/O
  https://vondra.me/posts/the-real-cost-of-random-io/
- Where do the performance cliffs come from?
  https://postgres.fm/episodes/performance-cliffs
  https://www.youtube.com/watch?v=UzdAelm-QSY
  https://www.youtube.com/watch?v=j0ISi1KVulU
- Fun and weirdness with SSDs
  https://vondra.me/posts/fun-and-weirdness-with-ssds/