Page Costs & the Buffer Cache

Robert Haas

where did you come from?

Everything Is On Disk

- With minor exceptions, PostgreSQL plans your query as if everything is on disk.

- Exception 1: effective_cache_size accounts for the possibility that **the same index scan** might hit the same page more than once.
- Exception 2: We disregard the cost of accessing non-leaf btree index pages.

Everything Is In Memory

- If your whole database fits in memory, you can model this by drastically reducing seq_page_cost and random_page_cost.
- Default values 4.0 and 1.0, maybe use 0.1 or 0.05 for both.
- You could also raise the cpu_cost parameters, which would be the same thing but more confusing.

Some Things Are In Memory

- You're hosed.
- Your best option is to fool around and come up with some blended value for random_page_cost and seq_page_cost that seems to produce good plans. (2.0/1.0 ; 0.5/0.3 ; others?)
- But this is really oversimplified.

Solution Sketch

- Part 1: Use statistics to estimate what portion of the table is likely to be in memory.
  - Hard, lots of ways to do it – and all of them have significant flaws.
- Part 2: Modify the formulas that use seq_page_cost and random_page_cost to take into account the chances of hitting a cached page.
  - Not trivial, but there aren't many places to change (mostly costsize.c) or as many possible approaches.