

30 saniyede kazanan, 15 dakikada kaybeden index

Aynı tablo, Postgres 17, üç ölçüm.

Muhammet Şafak • 3 Ekim 2026



**30 saniyede
neredeyse eşit.
15 dakikada
ayrıldılar.**

Partial ve composite index otuz saniyede 11.537'ye 10.795 tps.

On beş dakikalık sürekli yükte composite hedefi tutamadı.

Bir tabloda iki küme

	Arşiv · done	Canlı küme · pending
KİM BAKAR	Kimse sorgulamaz	✓ Worker'lar arar, claim'ler kilitler
BOYUT	Silinmezse sonsuza büyür	✓ Kabaca sabit durur
ÖLÇÜLEN	n_live_tup 10.005.726	✓ 5.000 satır
POSTGRES	Canlı tuple: index, istatistik ve vacuum eşiği onu sayar	✓ Aynı muamele; ikisi ayırt edilmez

Otuz saniye

Dört strateji, üç ölü satır kademesi. Canlı küme her kademedede 5.000 satır.

Üç kademe, dört strateji

STRATEJİ	100K ÖLÜ SATIR	1M	10M	INDEX (10M)
index yok	2.001 tps	247 tps	7 tps	—
(status)	6.499 tps	6.417 tps	6.426 tps	66,1 MB
(status, created_at)	12.414 tps	11.202 tps	10.795 tps	310,4 MB
partial (created_at) WHERE pending	13.041 tps	11.707 tps	11.537 tps	7,6 MB

INDEX YOK

Index'siz kuyruk yok oluyor.

Ölü satırla birlikte çökmüyor: 2.001'den 7 tps'ye. 10 milyon satırın arasından 5.000 satırı sequential scan saniyede yedi kez bulabiliyor.

41x

ASIL FARK BOYUTTA

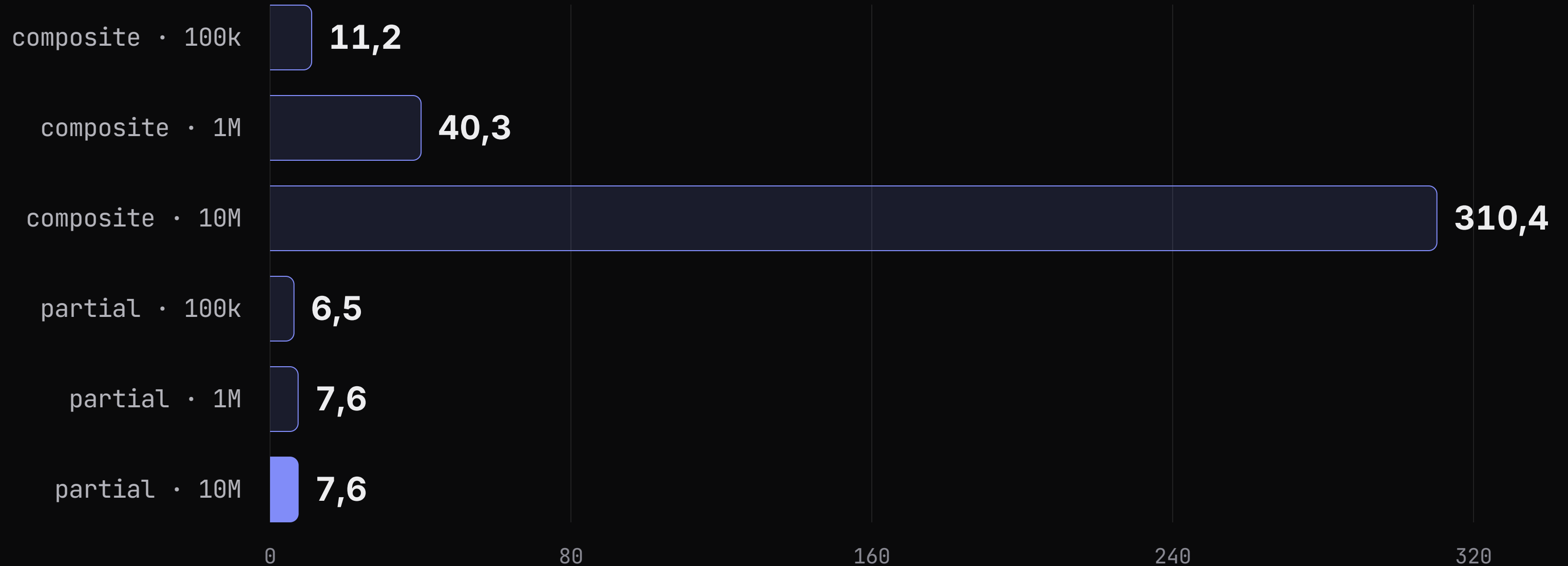
daha küçük index

10 milyon ölü satırda partial 7,6 MB, composite 310,4 MB. Throughput farkı yalnız %6,9: seçme sebebi hız değil, index'in tabloyla birlikte büyümemesi.

ÖLÜ SATIR ARTTIKÇA

Partial index tabloyla büyümüyor

index boyutu, MB



pg_relation_size, bench/run.sh

Planner

Aynı tablo, aynı index, aynı sorgu. Değişen tek şey planın türü.

10M ÖLÜ SATIR · AYNI SORGU, PARAMETREYLE

Generic plan u urumu

INDEX · PLAN	TPS	GE�İKME	PLAN
partial · custom	11.752	0,68 ms	Index Scan
partial · generic	7	1.113 ms	Sort → Seq Scan
composite · custom	11.105	0,72 ms	Index Scan
composite · generic	11.416	0,70 ms	Index Scan

Yaklaşık 1.635 kat

**NOT**

Generic plan \$1'in ne olduğunu bilmeden kurulur. status = \$1'in 'pending' olduğunu kanıtlayamaz, partial index'i eler.

**İPUÇU**

Composite etkilenmez: kanıtlanacak bir predicate'i yok, status index'in içindeki bir kolon.

**UYARI**

Uçuruma düşmek için force_generic_plan gerekti. Bu bir üretim ayarı değil.

Kırk çalıştırma, kırk custom plan



Bu koşullarda geçmedi



DÜZELTME

Önceki kaydımda 'Isındıkça bozulur' demiştim. Ölçüldüğünde yanlış çıktı; kırk çalıştırmanın kırkı custom plan kaldı.



NEDEN

En olası açıklama: partial index'i kullanamayan generic planın tahmini maliyeti yüksek çıkıyor. Plan maliyetleri kaydedilmedi.



BEDEL

Partial her çalıştırmada yeniden planlanır. Bu ölçekte fark ölçülemedi: 0,20 ms'ye 0,21 ms.

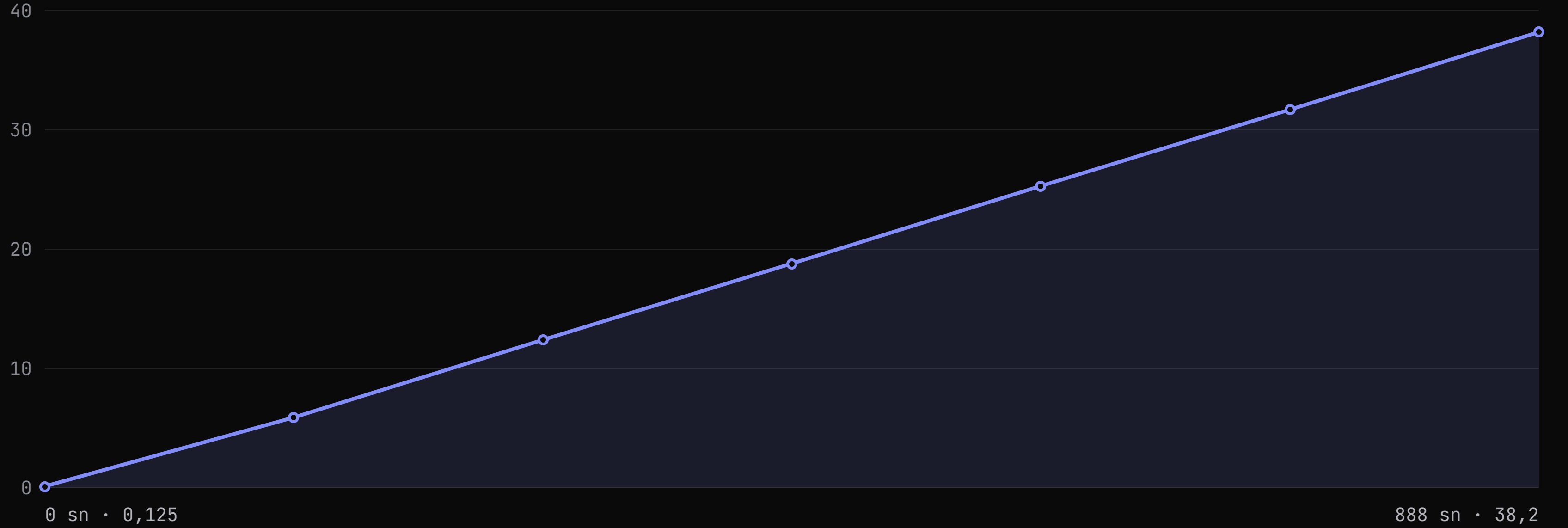
Şimdi 15 dakika

Sürekli churn: saniyede 2.000 iş, varsayılan
autovacuum ayarları.

PARTIAL · ~305 KAT

Canlı küme sabit, index büyüyor

index boyutu, MB

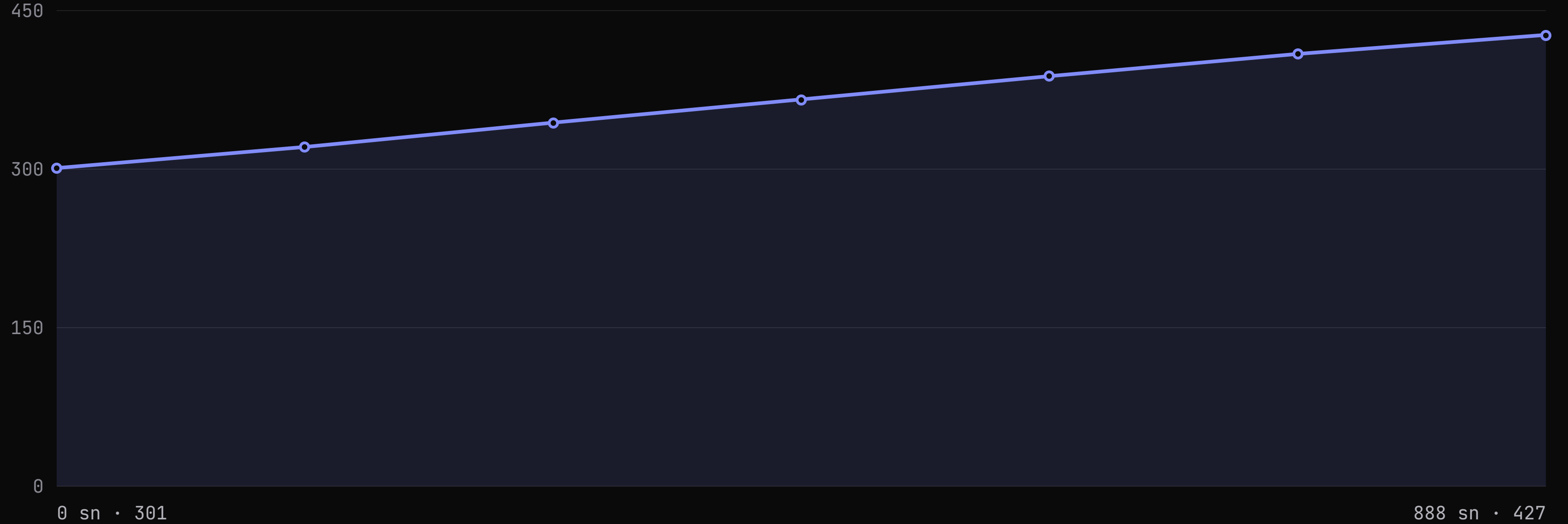


pg_relation_size, 15 saniyede bir örnek, bench/endurance.sh. pending→done geçişi vacuum'a kadar ölü girdi bırakır.

COMPOSITE · %42, +126 MB

Composite daha az oranla, daha çok MB şişti

index boyutu, MB



pg_relation_size, 15 saniyede bir örnek, bench/endurance.sh

HEDEF 2.000 İŞ/SN · 900. SANİYE

Şişen index sürdürülebilir yükü taşımıyor

	composite	partial
BAŞLANGIÇ	2.000 tps · 0,52 ms	✓ 2.024 tps · 1,7 ms
900. SANİYE	1.204 tps · 61.533 ms	✓ 2.016 tps · 4.906 ms
BEKLEYEN İŞ	5.000 → 126.024	✓ 5.003 → 14.364
SONUÇ	Hedefi tutamadı	✓ Hedefi tuttu

İki not



Gecikme pgbench'in zamanlama gecikmesini içeriyor: hedef hıza yetişemeyen istemcinin borcu buraya yazılıyor.



Fark sürdürülebilir yükte açılıyor: 427 MB'a şişen index artık belleğe sığmıyor, her tarama diske gidiyor.

Autovacuum hiç gelmedi

1,75

Milyon ölü satır

15 dakikada 1.753.949

0

Autovacuum

Bir kez bile koşmadı

≈ 2

Eşik, milyon satır

$50 + 0,2 \times 10.005.000 \approx 2.001.050$

17

Dakika, tahmini aralık

Bu iş yüküne özgü; vacuum maliyeti ölçülmedi

ÖNERİ

Eşiği tablodan koparın

Bu ayarı ölçmedim. 50.000 bir başlangıç noktası; kendi throughput'unuza göre ayarlayın.

jobs.sql

```
1 ALTER TABLE jobs SET (  
2     autovacuum_vacuum_scale_factor = 0,  
3     autovacuum_vacuum_threshold    = 50000  
4 );  
5  
6 CREATE INDEX CONCURRENTLY  
7     jobs_pending_created_at  
8     ON jobs (created_at)  
9     WHERE status = 'pending';
```

Sor Bakalım cevabındaki iddialar, ölçülünce

- ✓ Partial index küçük kalır: 10M'de 7,6 MB
- ✓ (status) index'inden üstün: 11.537'ye 6.426 tps
- ✓ ORDER BY kolonu index'e girmeli
- ✓ Parametreyle planner partial'ı seçemeyebilir: doğru ama dar
- ✓ Churn ve bloat: ilk kayıta ölçülmedi, sonradan ölçüldü



Koşulu bulun

github.com/muhammetsafak/pg-queue-bench

Bir tavsiyeyi ölçmenin değeri onu doğrulamak değil. Değeri, doğru bir cümlenin hangi koşulda tersine döndüğünü bulmakta.

