Zaawansowane Bazy Danych - Zadanie 4

Piotr Karpiński

11 stycznia 2021

Procedura testowa

Wszystkie dane na temat czasów wykonania zapytań przetrzymywałem w bazie danych - zarówno dla redisa jak i postgresa. Czasy modyfikacji tabel również były ustawiane po stronie serwera - to serwer zajmował się uzyskaniem timestampu. Taki sposób pomiaru pozwolił na wyeliminowanie problemu, w którym czas wstawienia do bazy zmierzony przez klienta mógł znacząco się różnić z rzeczywistym czasem aktualizacji tabeli - chociażby przez opóźnienie spowodowane komunikacją miedzy bazą a klientem.

Pojedynczy eksperyment składa się z dodania przez każdy z n procesów typu pierwszego 1000 wpisów, gdzie między dodaniem kolejnego wpisu proces usypiał na t sekund. Następnie wpisy zostają obsłużone przez n procesów typu drugiego i n procesów typu trzeciego. Przeprowadziłem eksperymennty dla $n \in \{1..16\}$ i $t \in \{0.001, 0.003, 0.006, 0.009\}$.

Obydwie bazy danych działały pod dockerem z flagami `--cpus="1.0" --memory="512m"`

Eksperymenty zostały przeprowadzone na komputerze z procesorem intel i5-4690k 3.50GHz oraz 8GB 2133MHz.

Działanie poszczególnych procesów

Proces 1

Proces ten po wstawieniu zalążka reklamy wysyła powiadomienie do losowo wybranego procesu 3, po czym wysyła powiadomienie do procesu 2, zawierając w powiadomieniu informację o tym ,który z procesów 3 otrzymał bieżącą reklamę do obsłużenia.

Proces 2

Proces drugi po otrzymaniu informacji od procesu pierwszego, uzupełnia dane w bazie, po czym wysyła powiadomienie do procesu trzeciego zajmującego się aktualnie obsługiwanym wpisem w bazie.

Proces 3

Proces trzeci po otrzymaniu informacji od procesu pierwszego może wyemitować reklamę od razu z prawdopodobieństwem 0.1, albo czeka na informację z procesu drugiego. Jeżeli proces postawił czekać, zapamiętuję `id` wpisu i czeka na powiadomienie od procesu drugiego. Po otrzymaniu dodatkowych informacji od procesu drugiego emituje reklamę z prawdopodobieństwem 0.7, w przeciwnym przypadku reklama nie jest emitowana.

Postgres

Tabela z danymi, w której przechowywałem dane wyglądała tak:

```
create table advert (
    id          serial primary key,
    cookie      text not null,
    ip          inet not null,
    city        text,
    country     text,
    time_in     timestamptz default current_timestamp,
    time_mid    timestamptz,
    time_end    timestamptz,
    emmit_type  int
);
```

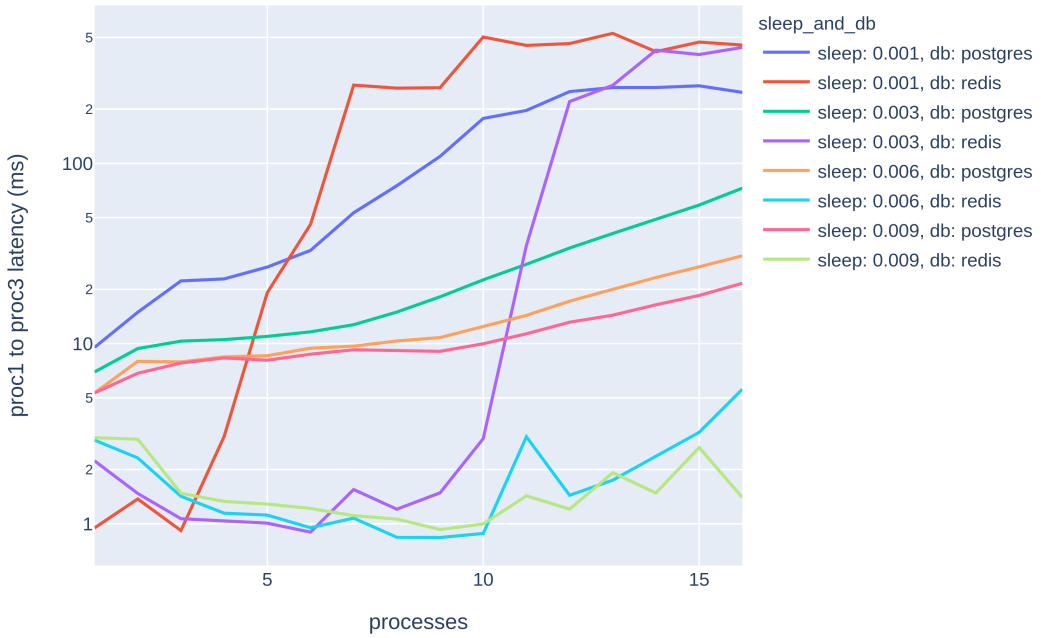
Do komunikacji między procesami użyłem `notify` i `listen`.

Redis

W przypadku Redisa dane przechowywałem w hashmapie, a każdy z kluczy przechowywałem w secie. Do komunikacji miedzy procesami skorzystałem z `publish` i `subscribe`.

Analiza wyników

Porównanie czasu obsługi w obrębie jednej bazy

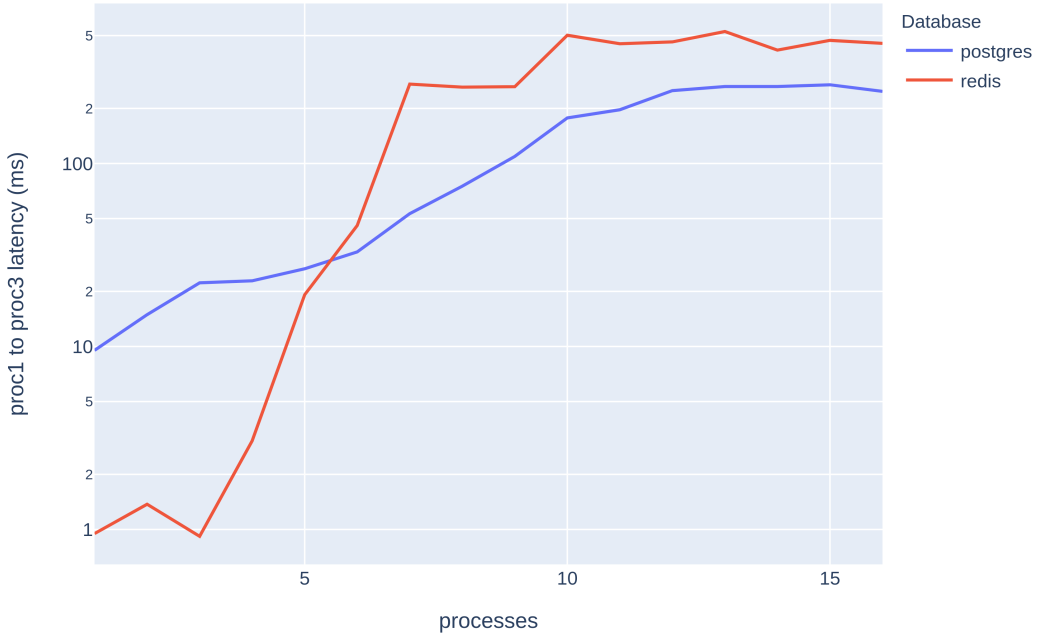


Rysunek 1: Średni czas obsługi reklamy

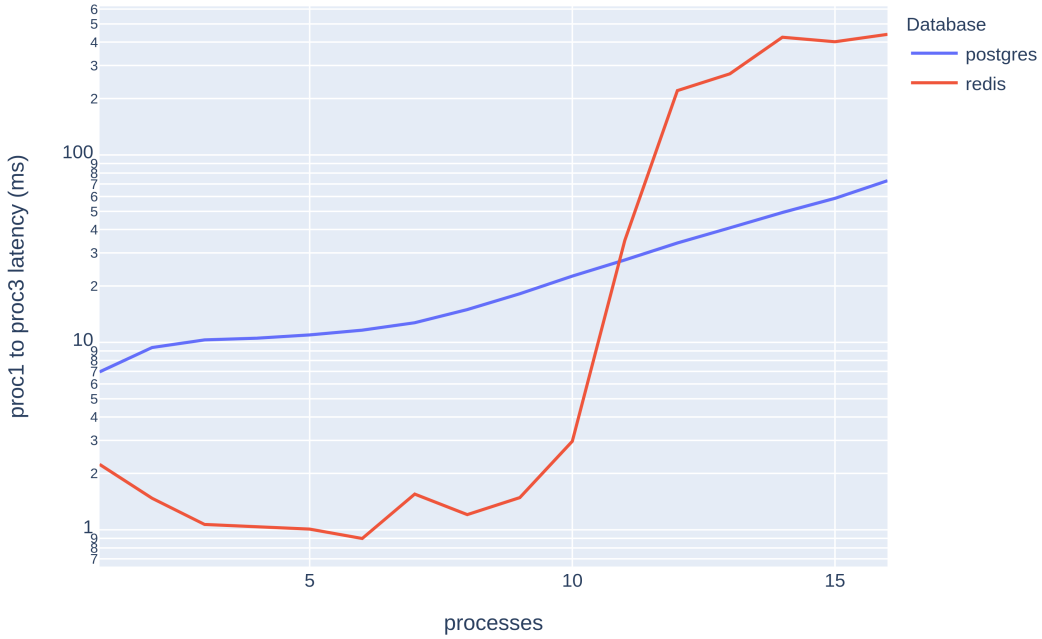
Wzrost czasu wykonania w zależności od liczby procesów jest dosyć wolny i stabilny w przypadku postgresa.

Wzrost czasu wykonania w zależności od liczby procesów jest dużo bardziej gwałtowny jeżeli używamy redisa. Jak widać na wykresie czasy obsługi reklamy są dosyć stabilne, do pewnego momentu, potem

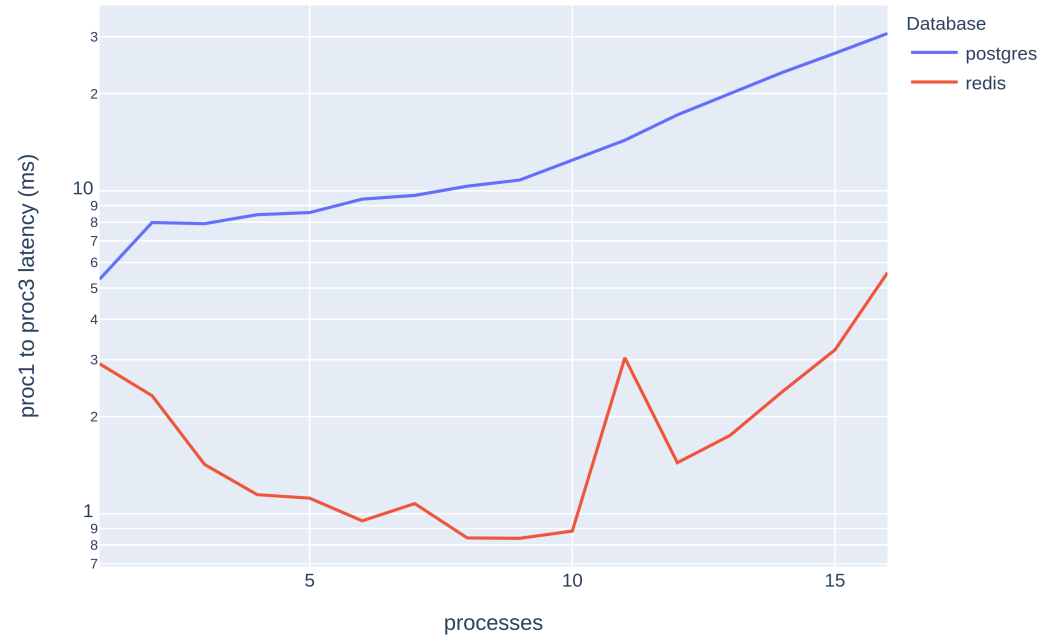
opóźnienie eksploduje. Warto zauważyć, ze przed punktem degradacji wydajności, obsługa reklamy przez redisa zajmuje mniej niż 5ms, natomiast dla postgresa są to liczby nieosiągalne przy żadnym ustawieniu.



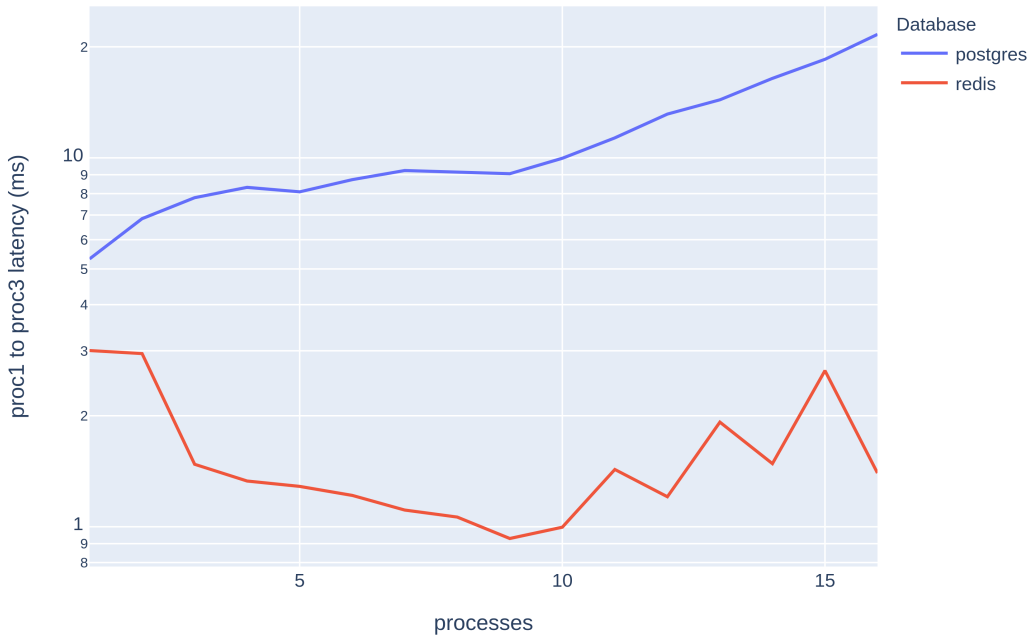
Rysunek 2: Porównanie średniego czasu obsługi reklamy w zależności od bazy danych dla t=0.001



Rysunek 3: Porównanie średniego czasu obsługi reklamy w zależności od bazy danych dla t=0.003



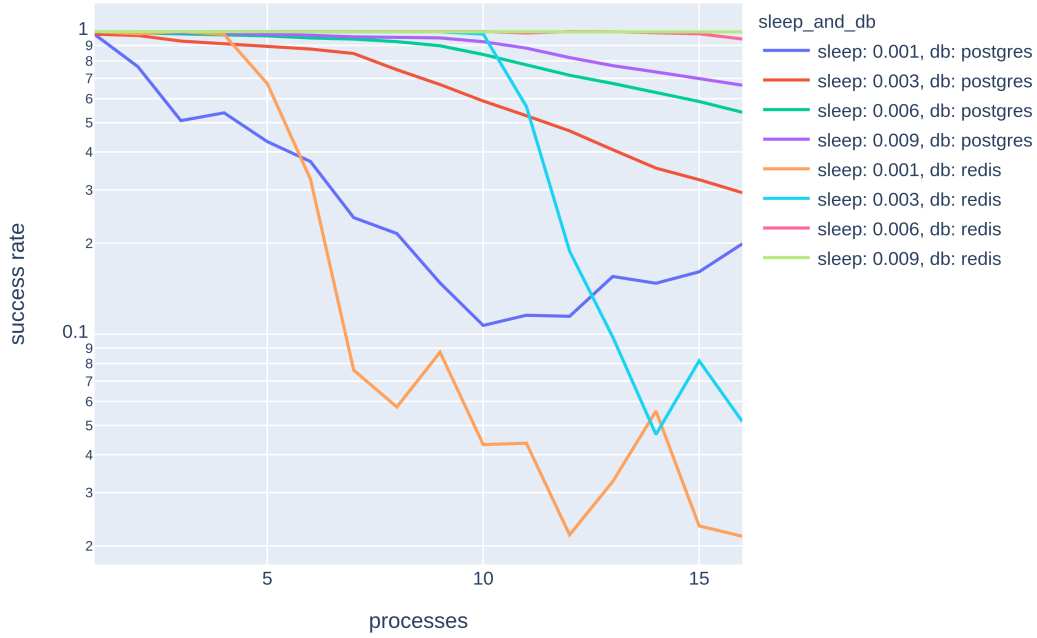
Rysunek 4: Porównanie średniego czasu obsługi reklamy w zależności od bazy danych dla t=0.006



Rysunek 5: Porównanie średniego czasu obsługi reklamy w zależności od bazy danych dla t=0.009

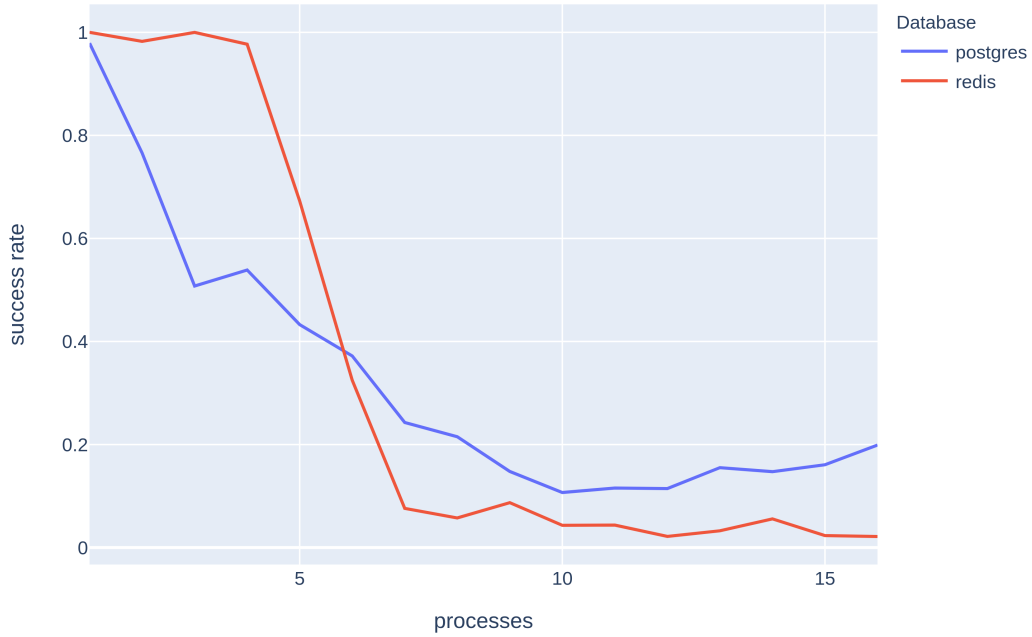
Redis zazwyczaj jest dużo szybszą bazą niż postgres, ale kiedy obciążenie jest bardzo wysokie, to postgres oferuję szybsza obsługę reklam. Szczególnie jest to widoczne dla $t = 0.001$ i $t = 0.003$.

Prawdopodobieństwo wyemitowania reklamy poniżej 20ms

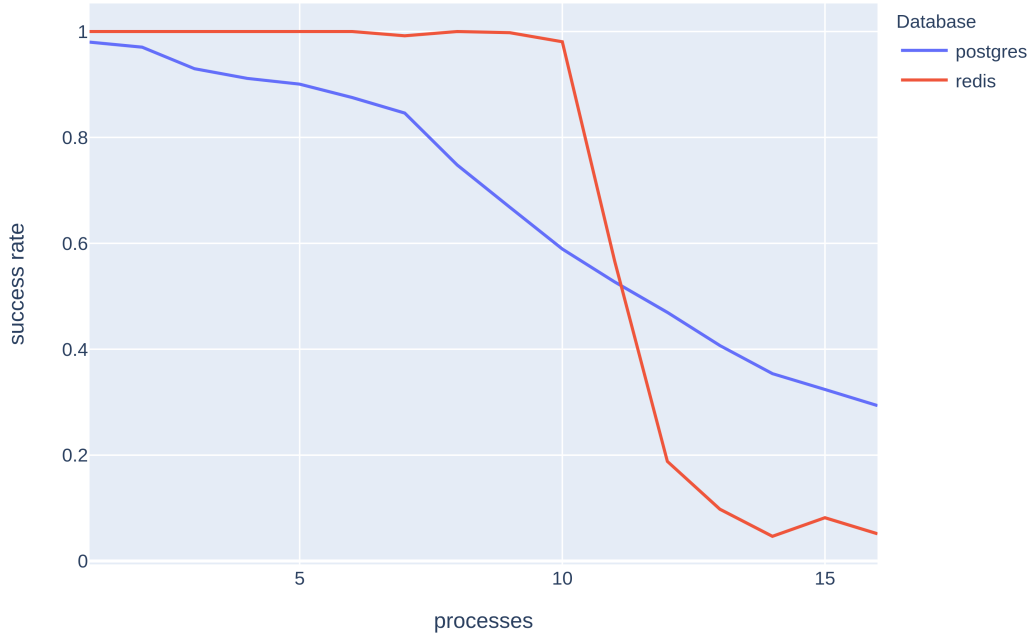


Rysunek 6: Porównanie prawdopodobieństwa wyemitowania reklamy poniżej 20ms

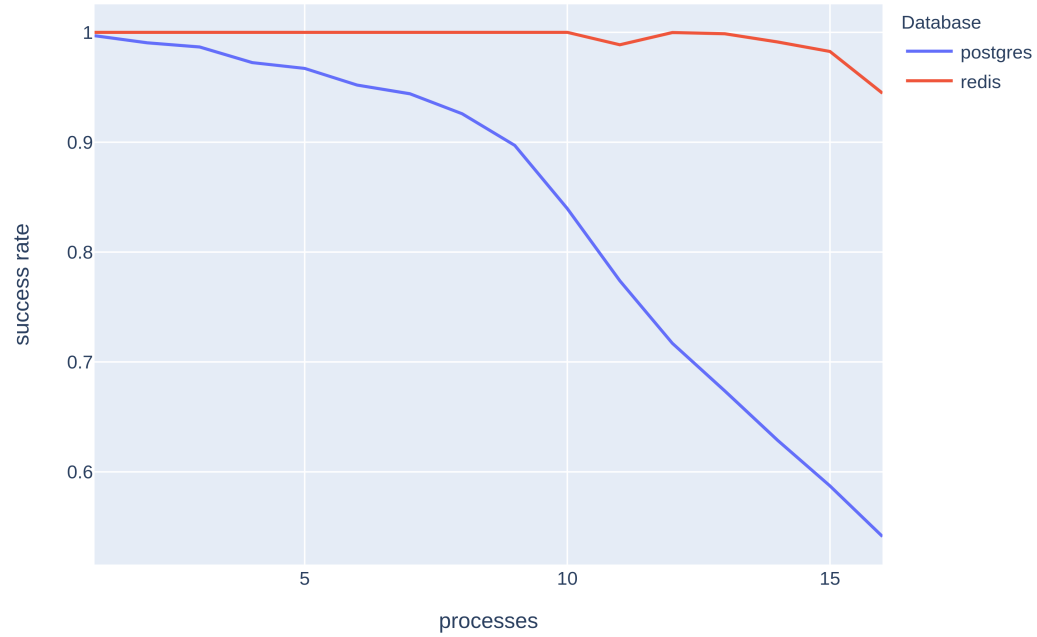
Podobnie jak dla czasu wykonania, prawdopodobieństwo wyemitowania reklamy dla postgresa spada wolno i przewidywalnie a dla redisa, obserwujemy gwałtowny skok.



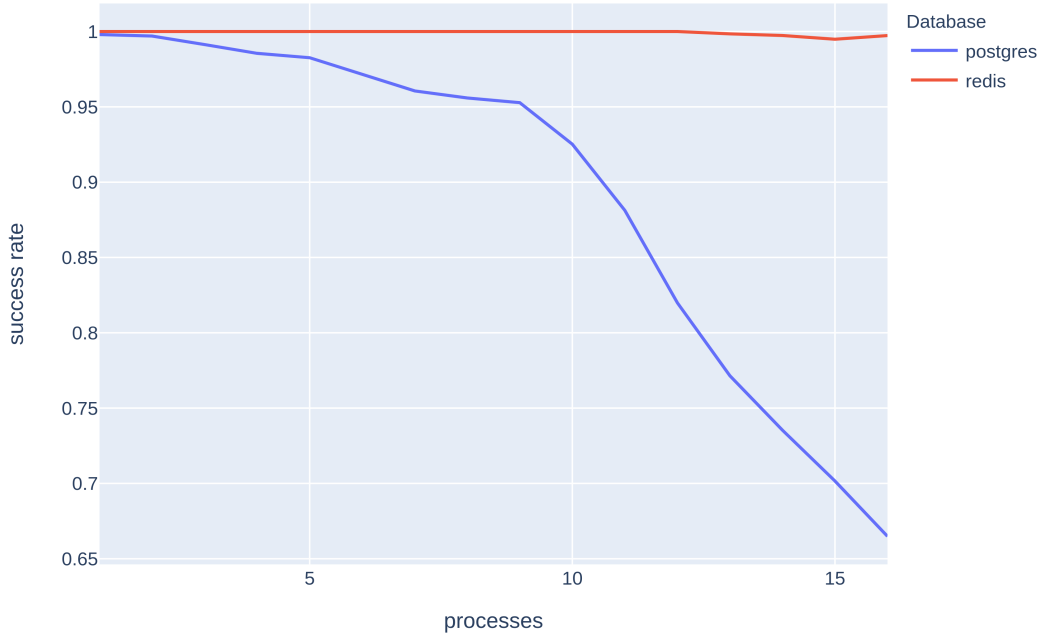
Rysunek 7: Porównanie prawdopodobieństwa wyemitowania reklamy poniżej 20ms w zależności od bazy danych dla t = 0.0001



Rysunek 8: Porównanie prawdopodobieństwa wyemitowania reklamy poniżej 20ms w zależności od bazy danych dla t = 0.0003



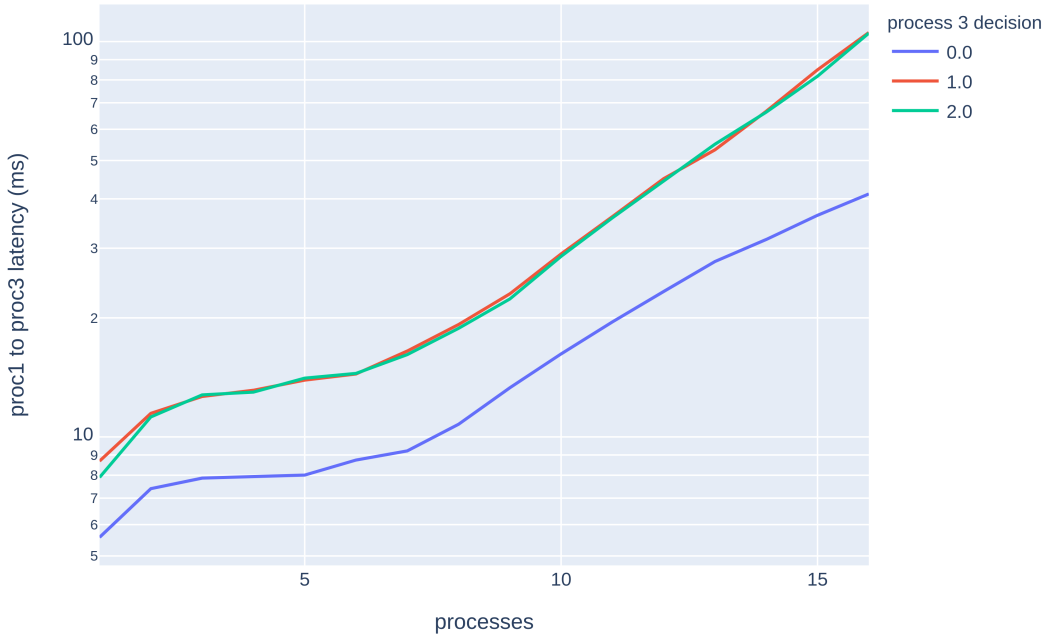
Rysunek 9: Porównanie prawdopodobieństwa wyemitowania reklamy poniżej 20ms w zależności od bazy danych dla t = 0.0006



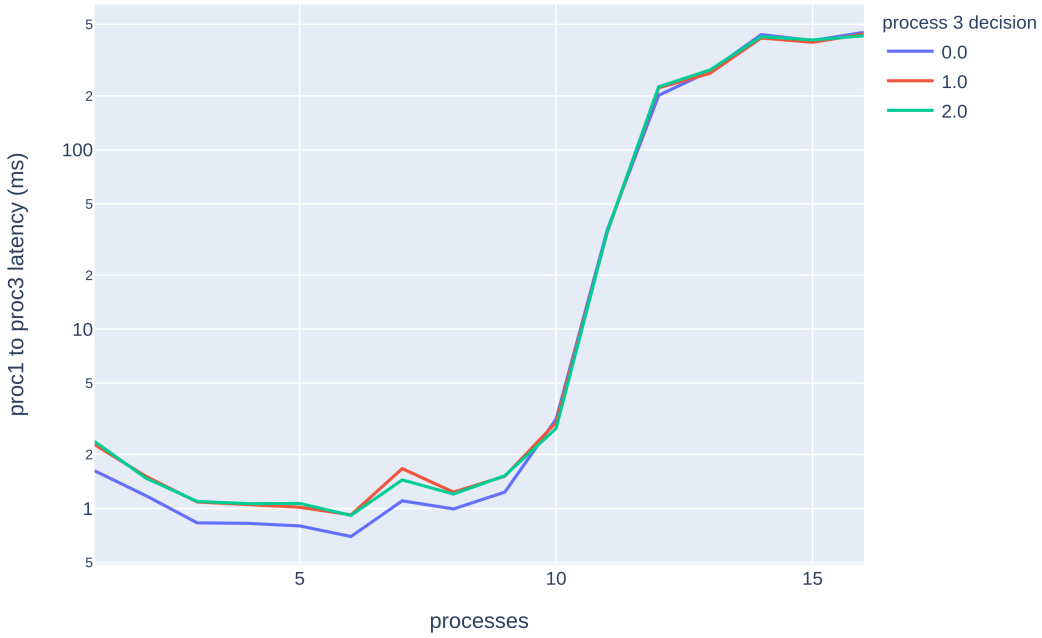
Rysunek 10: Porównanie prawdopodobieństwa wyemitowania reklamy poniżej 20ms w zależności od bazy danych dla t = 0.0009

Jeżeli obciążenie bazy ma być bardzo wysokie, lepszym rozwiązaniem może być użycie postgresa zamiast redisa, jeżeli zależy nam jedynie na obsłudze zapytań poniżej 20ms a sama szybkość odpowiedzi w przy normalnym obciążeniu nie jest sprawą drugorzędna.

Szybkość obsługi reklamy w zależności od decyzji procesu trzeciego



Rysunek 11: Czas obsługi reklamy dla postgresa w zależności od decyzji o wyemitowaniu reklamy - (0 - emituj natychmiast, 1 - emituj po otrzymaniu danych od procesu drugiego, 2 - nie emituj)



Rysunek 12: Czas obsługi reklamy dla redisa w zależności od decyzji o wyemitowaniu reklamy - (0 - emituj natychmiast, 1 - emituj po otrzymaniu danych od procesu drugiego, 2 - nie emituj)

Jeżeli proces trzeci potrzebuje danych z procesu drugiego, czas obsługi wydłuża się dwukrotnie w porównaniu z procesem dla którego informacje z procesu pierwszego są wystarczające. Dla postgresa takie zachowanie utrzymuje bez różnicy na liczbę procesów połączonych z bazą. Przy dużym obciążeniu, czas obsługi zapytań przez redisa jest taki bez różnicy jaką decyzje podejmie proces trzeci.

Podsumowanie

Wbrew oczekiwaniom redis nie jest zawsze bazą szybszą od postgresa. Postgres zachowuje się dużo bardziej przewidywalnie od redisa i może być lepszym wyborem, jeżeli chcemy rozwiązania wolniejszego ale stabilniejszego.