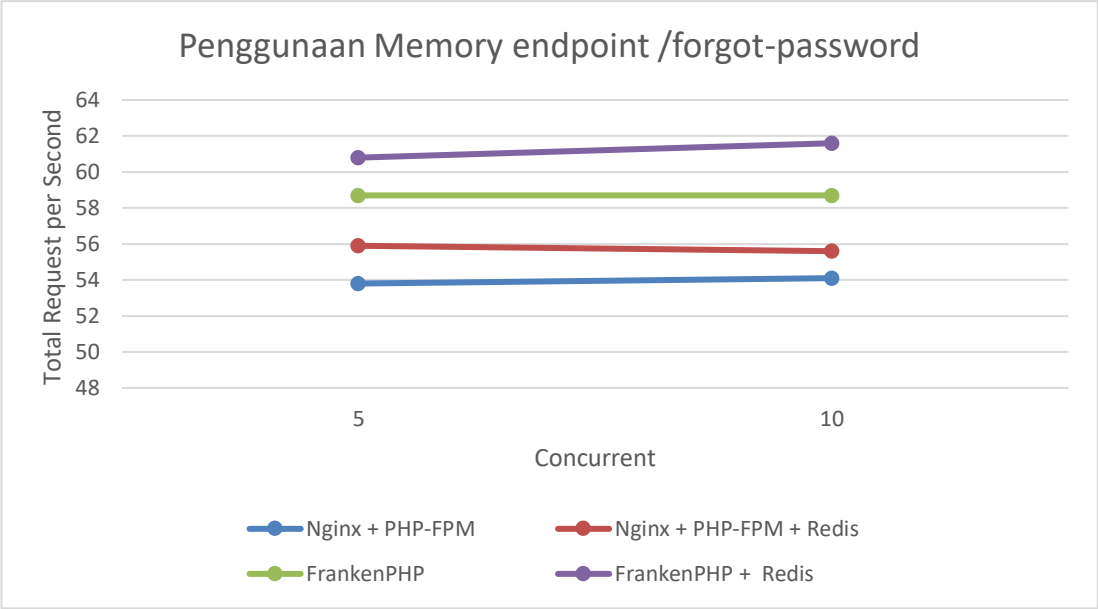
FrankenPHP cenderung membutuhkan alokasi memori tambahan untuk menjaga performa proses asynchronous dan peningkatan concurrency. Jadi, peningkatan efisiensi eksekusi dibayar dengan konsumsi memori yang lebih tinggi.

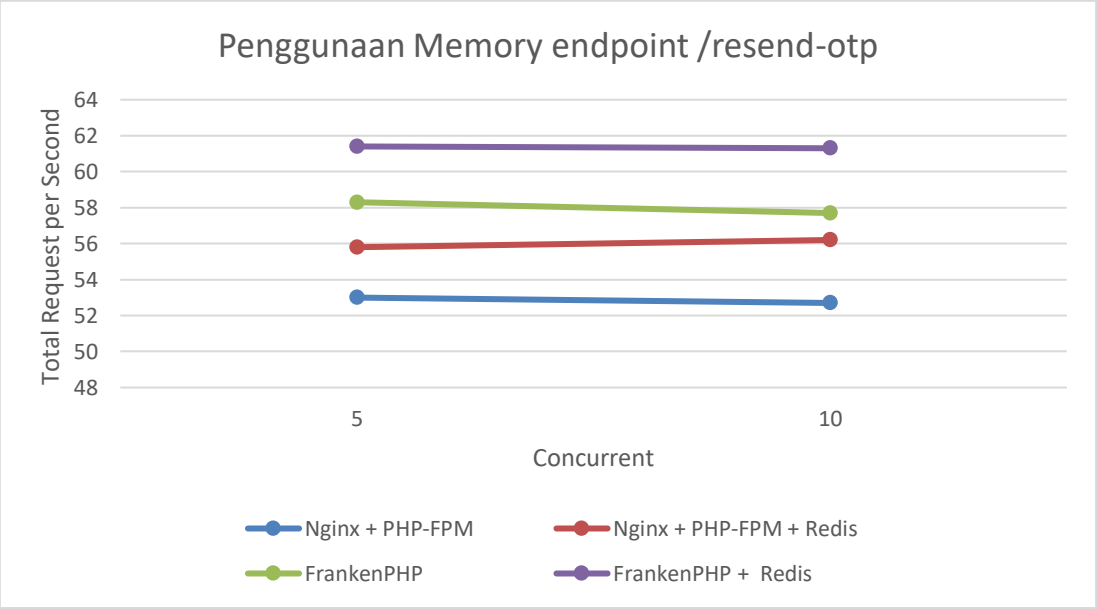


Gambar 4. 52. Grafik Penggunaan Memory Pada Endpoint /forgot-password

Menurut data pada tabel 4.70 dan grafik pada gambar 4.52 menunjukkan penggunaan memory pada endpoint /forgot-password dalam dua skenario concurrent (5 dan 10), dengan empat kondisi: sebelum dioptimasi, setelah dioptimasi dengan *queue*, dengan FrankenPHP, dan kombinasi FrankenPHP dan *queue*. Berbeda dengan hasil pengujian pada CPU, pada aspek penggunaan memory justru terjadi peningkatan setelah optimasi dilakukan, baik melalui queue maupun penggunaan FrankenPHP.

Secara rata-rata, penggunaan memory sebelum dioptimasi berada pada kisaran 53,8% hingga 54,1%. Setelah dioptimasi dengan queue, terjadi sedikit peningkatan menjadi 55,6% hingga 55,9%, atau naik sekitar 3% hingga 4%. Penggunaan FrankenPHP saja meningkatkan konsumsi memory menjadi 58,7%, naik sekitar 7% hingga 9% dari kondisi awal. Sementara itu, kombinasi antara FrankenPHP dan queue memberikan peningkatan paling besar dengan konsumsi memory mencapai 60,8% hingga 61,6%, yaitu meningkat sekitar 12% hingga 15% dibandingkan sebelum dilakukan optimasi.

Dengan demikian, meskipun optimasi dengan queue dan FrankenPHP sangat efektif dalam menurunkan penggunaan CPU, terdapat trade-off berupa peningkatan penggunaan memory. Peningkatan ini dapat dianggap sebagai konsekuensi dari proses asynchronous dan performa server yang lebih tinggi, yang memanfaatkan lebih banyak resource memory untuk menjaga kecepatan dan efisiensi proses secara keseluruhan.



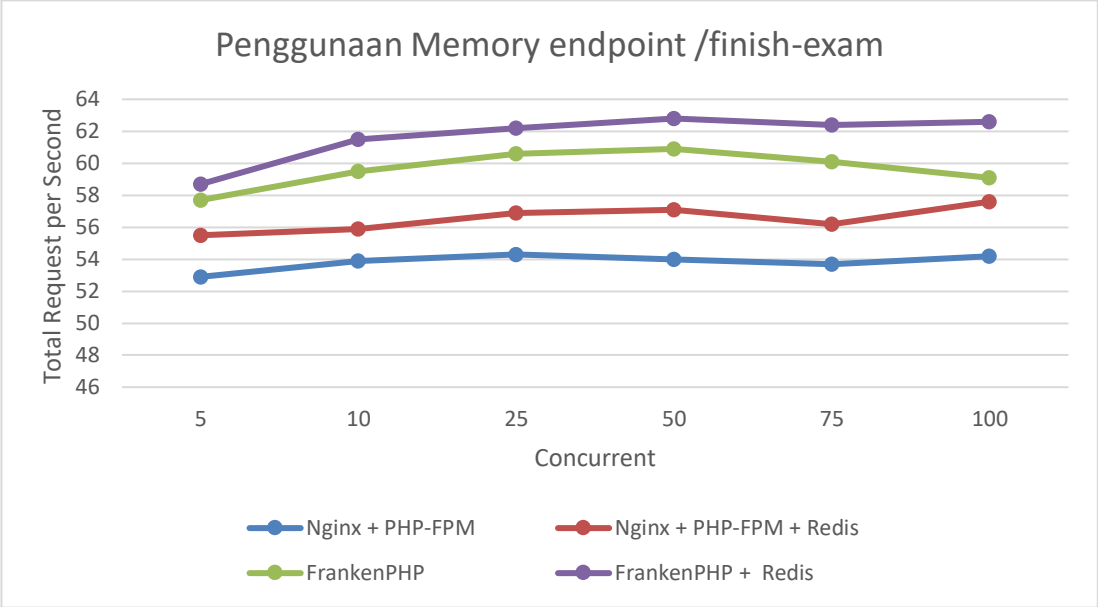
Gambar 4. 53. Grafik Penggunaan Memory Pada Endpoint /resend-otp

Data pada tabel 4.70 dan grafik pada gambar 4.53 penggunaan memori pada endpoint /resend-otp menunjukkan adanya peningkatan konsumsi memori setelah dilakukan beberapa metode optimasi. Sebelum dioptimasi, penggunaan memori berada pada kisaran 52,7% hingga 53%. Setelah diterapkan queue, penggunaan memori justru mengalami kenaikan sebesar sekitar 5,7% hingga 7,3%, yaitu menjadi 55,8% hingga 56,2%. Ini menunjukkan bahwa meskipun queue mampu membantu pengelolaan beban proses di CPU, ada konsekuensi tambahan pada sisi penggunaan memori.

Penggunaan FrankenPHP sendiri juga menunjukkan kenaikan memori dibandingkan sebelum dioptimasi, yaitu meningkat menjadi 57,7% hingga 58,3%, atau naik sekitar 9,5% hingga 10%. Hal ini dapat dimaklumi mengingat

FrankenPHP dirancang untuk kinerja tinggi dan dapat menyimpan lebih banyak konteks atau proses dalam memori untuk kecepatan eksekusi.

Peningkatan paling signifikan adalah pada kombinasi FrankenPHP dan queue, di mana penggunaan memori meningkat hingga 61,3% hingga 61,4%. Artinya, dibandingkan dengan kondisi awal, terjadi kenaikan memori antara 15,8% hingga 16%. Secara keseluruhan, dapat disimpulkan bahwa meskipun kombinasi optimasi ini efektif di sisi performa, khusus pada endpoint /resend-otp, trade-off-nya adalah penggunaan memori yang lebih tinggi, dengan kenaikan berkisar antara 5% hingga 16%, tergantung jenis optimasi yang diterapkan.



Gambar 4. 54. Grafik Penggunaan Memory Pada Endpoint /finish-exam

Endpoint /finish-exam pada tabel 4.70 dan gambar 4.54 menunjukkan penggunaan memory pada endpoint /finish-exam dalam berbagai skenario concurrent dan iteration. Tidak seperti pada penggunaan CPU, optimasi yang dilakukan justru menunjukkan kenaikan penggunaan memory dibandingkan sebelum dioptimasi. Sebelum optimasi, penggunaan memory berkisar antara 52,9% hingga 54,3%. Setelah dioptimasi menggunakan queue, memory meningkat menjadi 55,5% hingga 57,6%, yang berarti ada kenaikan sebesar sekitar 4,9% hingga 6,3%.