JOURNAL OF INFORMATION & TECHNOLOGY
journal-inform-tech, 1(1), 2026, 9

Ambang Konsolidasi Tenant pada Arsitektur SaaS Multi-Tenant: Model Antrean M/M/c untuk Menyeimbangkan Latensi p95 dan Biaya Infrastruktur

Tenant Consolidation Thresholds in Multi-Tenant SaaS Architectures: An M/M/c Queueing Model for Balancing p95 Latency and Infrastructure Cost

Abdullah Wasian[1*]

[1] *Ianstudios Institute*
* Penulis korespondensi: Abdullah Wasian (admin@ianstudios.id)

Diterima: 2026-11-15 · Disetujui: 2026-11-15 · Terbit: 2026-11-14

ABSTRAK

Konsolidasi banyak tenant pada satu node adalah sumber utama efisiensi biaya aplikasi Software as a Service (SaaS) multi-tenant, tetapi menaikkan antrean dan memperpanjang ekor latensi. Penelitian ini menurunkan ambang konsolidasi, yaitu jumlah tenant terbanyak per node yang masih memenuhi target tingkat layanan (SLO) latensi persentil ke-95 (p95), memakai model antrean M/M/c dengan rumus Erlang C dan distribusi waktu tinggal eksak. Model divalidasi dengan simulasi kejadian diskrit (10 replikasi × 110.000 permintaan per skenario). Pada konfigurasi dasar (8 worker, waktu layanan rata-rata 40 ms, 2 permintaan/detik per tenant), 60 tenant per node menghasilkan utilisasi 60,0%, peluang menunggu 14,0%, latensi rata-rata 41,7 ms, dan latensi p95 122,3 ms. Ambang konsolidasi untuk SLO p95 200 ms adalah 90 tenant per node, dengan biaya Rp48.667 per tenant per bulan, 91,1% lebih rendah daripada isolasi penuh. Galat p95 model terhadap simulasi kurang dari 0,3% pada seluruh skenario. Hasil juga menunjukkan bahwa latensi rata-rata menyembunyikan lonjakan ekor: setelah utilisasi sekitar 80%, p95 naik jauh lebih cepat daripada rata-rata, dan bila waktu layanan 80 ms, SLO tidak dapat dicapai berapa pun jumlah worker-nya. Model, data, dan kode dibuka untuk diuji ulang pembaca melalui Lab Simulasi.

Kata kunci: SaaS multi-tenant; teori antrean; Erlang C; latensi ekor; perencanaan kapasitas; biaya cloud

ABSTRACT

Consolidating many tenants on one node is the main source of cost efficiency in multi-tenant Software as a Service (SaaS) applications, but it lengthens queues and the latency tail. This study derives the consolidation threshold, the largest number of tenants per node that still meets a 95th-percentile (p95) latency service level objective (SLO), using an M/M/c queueing model with the Erlang C formula and the exact sojourn-time distribution. The model is validated against discrete-event simulation (10 replications of 110,000 requests per scenario). In the base configuration (8 workers, 40 ms mean service time, 2 requests/s per tenant), 60 tenants per node yield 60.0% utilisation, a 14.0% waiting probability, 41.7 ms mean latency, and 122.3 ms p95 latency. The threshold for a 200 ms p95 SLO is 90 tenants per node, at Rp48,667 per tenant per month, 91.1% cheaper than full isolation. The p95 error against simulation is below 0.3% in all scenarios. Mean latency hides the tail surge beyond about 80% utilisation, and with an 80 ms service time the SLO cannot be met regardless of the number of workers. The model, data, and code are open for readers to re-run in the Simulation Lab.

1. Pendahuluan

Komputasi awan mengubah perangkat lunak menjadi layanan yang dibayar sesuai pemakaian. Definisi NIST menempatkan Software as a Service (SaaS) sebagai salah satu dari tiga model layanan, dengan pengumpulan sumber daya (resource pooling) sebagai karakteristik esensialnya (Mell & Grance, 2011). Armbrust dkk. (2010) menunjukkan bahwa keuntungan ekonomis cloud bersumber dari elastisitas dan multipleksing statistik: beban

DOI: https://doi.org/10.5555/jit.v1i1.d3a3ec8c © 2026 Penulis · CC BY 4.0

banyak pelanggan yang tidak serentak dapat dilayani perangkat keras yang lebih sedikit.

Pada SaaS, prinsip ini diwujudkan lewat arsitektur multi-tenant, yaitu satu instans aplikasi melayani banyak organisasi pelanggan (tenant) sekaligus. Bezemer dan Zaidman (2010) mencatat bahwa multi-tenancy memberi skala ekonomi pada perangkat keras dan pemeliharaan, sekaligus memunculkan risiko teknis baru. Krebs, Momm, dan Kounev (2012) menempatkan isolasi kinerja antar-tenant sebagai salah satu perhatian arsitektural utama, karena tenant yang berbagi sumber daya dapat saling memengaruhi waktu tanggap.

Pertanyaan praktis bagi penyedia SaaS adalah berapa tenant yang boleh ditempatkan pada satu node sebelum kualitas layanan turun. Semakin banyak tenant per node, semakin murah biaya per tenant, tetapi semakin tinggi utilisasi dan semakin panjang antrean. Dean dan Barroso (2013) menegaskan bahwa untuk layanan interaktif, yang menentukan pengalaman pengguna adalah ekor distribusi latensi, bukan rata-ratanya, dan ekor ini memburuk ketika utilisasi naik. Karena itu, perencanaan kapasitas yang hanya memakai latensi rata-rata cenderung terlalu optimistis.

Teori antrean menyediakan alat analitis untuk persoalan ini. Khazaei, Mišić, dan Mišić (2012) memodelkan pusat data cloud sebagai sistem antrean M/G/m/m+r untuk mengevaluasi waktu tanggap, dan Lorido-Botran, Miguel-Alonso, dan Lozano (2014) mencatat teori antrean sebagai salah satu keluarga teknik dalam penskalaan otomatis (auto-scaling). Namun, kajian-kajian tersebut berfokus pada pusat data atau penskalaan mesin virtual, bukan pada keputusan konsolidasi tenant dengan target persentil latensi.

Penelitian ini mengisi celah tersebut dengan tiga kontribusi. Pertama, menurunkan ambang konsolidasi tenant per node untuk SLO latensi p95 dari model M/M/c dengan distribusi waktu tinggal eksak. Kedua, memvalidasi model terhadap simulasi kejadian diskrit dan mengukur trade-off biaya terhadap isolasi penuh. Ketiga, membuka model, data, dan kode sehingga pembaca dapat mengubah parameter dan menguji batas klaim penelitian secara langsung.

2. Tinjauan Pustaka

2.1 Multi-Tenancy pada SaaS

Kabbedijk dkk. (2015) memetakan definisi multi-tenancy dari sudut pandang akademik dan industri, dan menyimpulkan bahwa ciri utamanya adalah berbagi sumber daya antar-pelanggan dengan tetap memberi kesan isolasi. Fehling, Leymann, dan Mietzner (2010) mengusulkan kerangka penempatan tenant yang optimal pada aplikasi cloud, sehingga penempatan tenant dipandang sebagai persoalan optimisasi dengan batasan kualitas layanan. Penelitian ini mengikuti sudut pandang tersebut, tetapi memakai ukuran kualitas yang lebih ketat, yaitu persentil latensi.

2.2 Model Antrean untuk Sistem Cloud

Harchol-Balter (2013) menyajikan model M/M/k untuk server farm dan pemakaiannya dalam penyediaan kapasitas, termasuk efek skala: server farm yang lebih besar dapat dijalankan pada utilisasi lebih tinggi dengan waktu tunggu yang sama. Model ini menjadi dasar penelitian ini. Khazaei dkk. (2012) memperluas analisis ke waktu layanan berdistribusi umum dan antrean berkapasitas terbatas, dengan konsekuensi solusi yang bersifat aproksimasi.

2.3 Latensi Ekor

Dean dan Barroso (2013) menunjukkan bahwa episode latensi tinggi yang jarang dapat mendominasi kinerja layanan berskala besar. Konsekuensinya, SLO layanan interaktif lazim dinyatakan dalam persentil (p95 atau p99). Penelitian ini memakai p95 sebagai kriteria utama dan melaporkan p99 sebagai pelengkap.

3. Metode

3.1 Model Sistem

Satu node SaaS dimodelkan sebagai sistem M/M/c: c worker identik melayani satu antrean FCFS bersama. Setiap tenant mengirim permintaan menurut proses Poisson berlaju λ, sehingga superposisi N tenant identik menghasilkan kedatangan Poisson berlaju Nλ. Waktu layanan berdistribusi eksponensial dengan rata-rata S (laju μ = 1/S). Asumsi tenant homogen dan waktu layanan eksponensial dipilih agar distribusi waktu tinggal dapat dihitung eksak; dampak asumsi ini dibahas di Bagian 5.

3.2 Ukuran Kinerja

Beban yang ditawarkan adalah a = Nλ/μ dan utilisasi per worker ρ = a/c, dengan syarat stabil ρ < 1. Peluang sebuah permintaan harus menunggu diberikan rumus Erlang C (Harchol-Balter, 2013):

$$C(c, a) = [a^c/c! \cdot 1/(1 - \rho)] / [\Sigma_{0}^{c-1} a^k/k! + a^c/c! \cdot 1/(1 - \rho)] \quad (1)$$

Latensi rata-rata (waktu tinggal = tunggu + layanan) adalah:

$$E[T] = C(c, a) / (c\mu - N\lambda) + 1/\mu \quad (2)$$

Pada M/M/c FCFS, waktu tunggu bersyarat menunggu berdistribusi eksponensial dengan laju θ = cμ − Nλ. Dengan konvolusi terhadap waktu layanan, fungsi survival waktu tinggal adalah:

$$P(T > t) = (1 - C)\cdot e^{-\mu t} + C\cdot(\theta\cdot e^{-\mu t} - \mu\cdot e^{-\theta t}) / (\theta - \mu) \quad (3)$$

Latensi p95 adalah nilai t yang memenuhi P(T > t) = 0,05 dan dihitung dengan metode bagi dua (bisection) hingga presisi di bawah 10^{-9} detik. Rumus Erlang C dihitung lewat rekursi Erlang B agar stabil secara numerik untuk c besar.

3.3 Ambang Konsolidasi dan Biaya

Ambang konsolidasi N_a didefinisikan sebagai N terbesar yang memenuhi p95(N) ≤ SLO. Karena p95 naik monoton terhadap N, N_a dicari dengan pencarian biner. Biaya per tenant per bulan adalah H × 730 / N, dengan H harga sewa node per jam dan 730 jam per bulan. Sebagai pembanding, skema isolasi penuh memberi setiap tenant satu node 1 worker dengan harga sebanding jumlah worker (H/c per jam).

3.4 Validasi dengan Simulasi Kejadian Diskrit

Model divalidasi dengan simulasi kejadian diskrit (discrete-event simulation, DES) M/M/c FCFS yang ditulis independen dari rumus analitis. Untuk N = 30, 45, 60, 75, dan 90 dijalankan 10 replikasi ber-seed, masing-masing 120.000 permintaan dengan 10.000 permintaan pertama dibuang sebagai periode pemanasan. Interval kepercayaan 95% dihitung dengan distribusi t (9 derajat bebas).

3.5 Parameter

Tabel 1 merangkum nilai dasar parameter. Nilai-nilai ini merepresentasikan layanan API interaktif berukuran sedang dan dapat diubah pembaca di Lab Simulasi.

Tabel 1. Parameter dasar model

Parameter	Simbol	Nilai dasar	Rentang di Lab Simulasi
Tenant per node	N	60	1–150
Laju permintaan per tenant	λ	2 permintaan/detik	0,2–6
Waktu layanan rata-rata	S	40 ms	5–120 ms
Worker per node	c	8	1–32

Parameter	Simbol	Nilai dasar	Rentang di Lab Simulasi
Harga node	H	Rp6.000/jam	Rp1.000–30.000
SLO latensi	p95	200 ms	tetap

3.6 Ketersediaan Kode dan Reproduksi

Seluruh perhitungan memakai Python 3 tanpa pustaka eksternal untuk model dan simulasi (model_saas.py, validasi_des.py, analisis.py), serta matplotlib untuk grafik (gambar.py). Data hasil (sapuan_tenant.csv, validasi_des.csv, sensitivitas.csv, perbandingan_biaya.csv) diunggah bersama naskah beserta sidik SHA-256-nya. Fungsi simulate(p) pada model_saas.py dapat dijalankan pembaca di browser melalui Lab Simulasi.

4. Hasil

4.1 Validasi Model

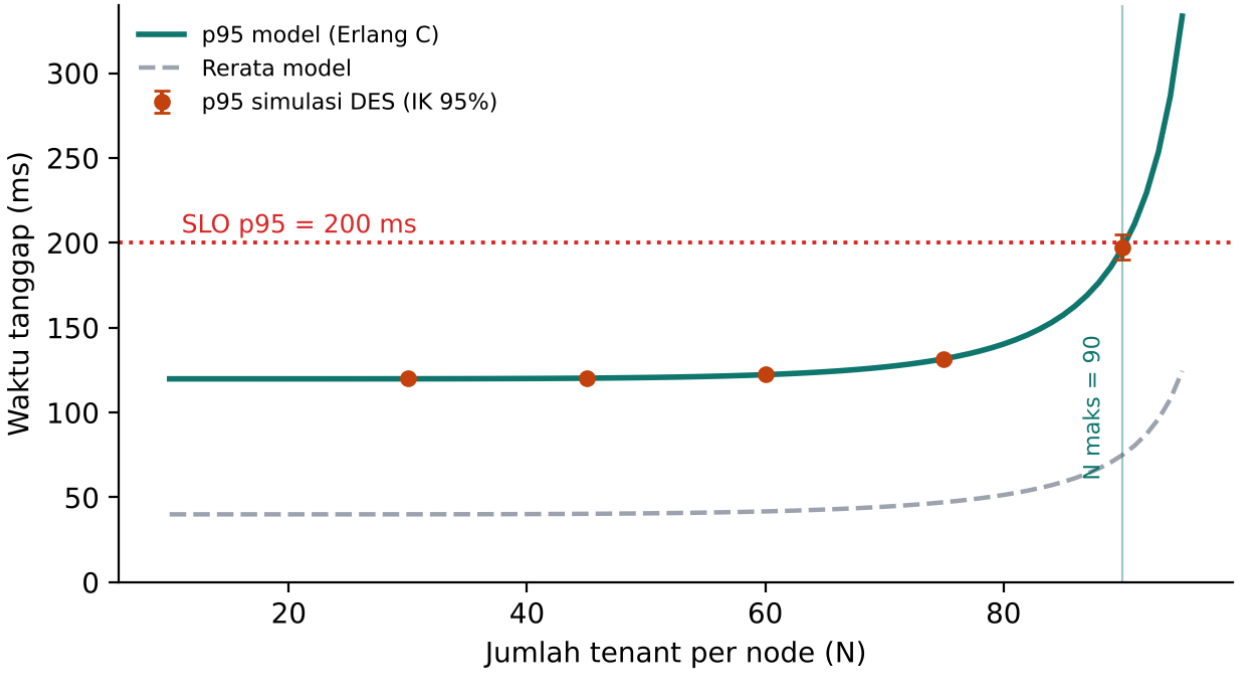
Tabel 2 membandingkan latensi rata-rata dan p95 hasil model dengan DES. Selisih p95 berkisar 0,06–0,29%, dan nilai model selalu berada di dalam interval kepercayaan 95% hasil simulasi. Lebar interval membesar pada N = 90 (utilisasi 90%), sesuai dengan meningkatnya variabilitas antrean pada beban tinggi.

Tabel 2. Validasi model analitis terhadap simulasi kejadian diskrit (10 replikasi)

N	Utilisasi	Rerata model (ms)	Rerata DES (ms)	p95 model (ms)	p95 DES (ms)	Galat p95
30	30%	40,0	40,0 ± 0,1	119,9	119,9 ± 0,5	0,06%
45	45%	40,3	40,3 ± 0,1	120,2	120,0 ± 0,4	0,20%
60	60%	41,7	41,8 ± 0,1	122,3	122,4 ± 0,4	0,13%
75	75%	47,1	47,0 ± 0,3	131,6	131,3 ± 0,7	0,29%
90	90%	75,1	75,8 ± 2,1	197,0	197,5 ± 7,6	0,26%

4.2 Pengaruh Jumlah Tenant terhadap Latensi

Gambar 1 dan Tabel 3 memperlihatkan bahwa latensi hampir tidak berubah hingga sekitar 50 tenant per node, karena peluang menunggu masih kecil. Pada nilai dasar N = 60, utilisasi mencapai 60,0%, peluang menunggu 14,0%, latensi rata-rata 41,7 ms, latensi p95 122,3 ms, dan p99 186,7 ms. Setelah utilisasi sekitar 80%, p95 naik tajam: dari 140,3 ms pada N = 80 menjadi 197,0 ms pada N = 90 dan 333,5 ms pada N = 95. Ambang konsolidasi untuk SLO p95 200 ms adalah 90 tenant per node.



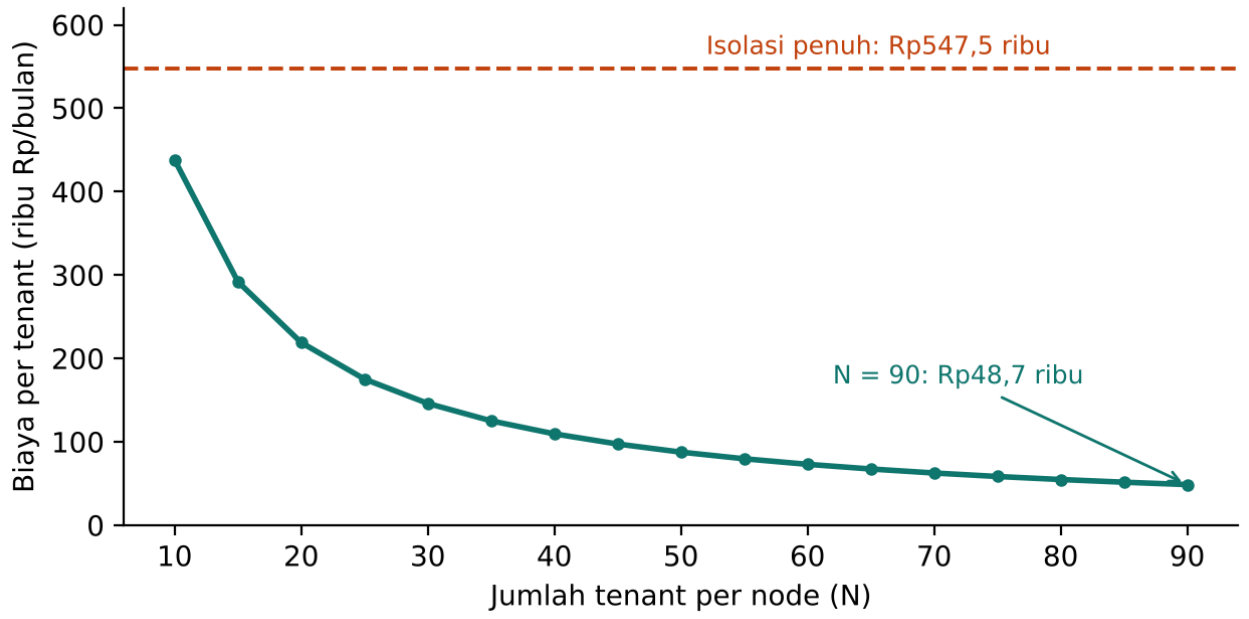
Gambar 1. Latensi rata-rata dan p95 terhadap jumlah tenant per node; titik oranye adalah hasil simulasi kejadian diskrit dengan interval kepercayaan 95%.

Tabel 3. Ukuran kinerja pada konfigurasi dasar untuk beberapa jumlah tenant

N	Utilisasi	Peluang menunggu	Rerata (ms)	p95 (ms)	p99 (ms)	Memenuhi SLO
30	30,0%	0,4%	40,0	119,9	184,2	Ya
60	60,0%	14,0%	41,7	122,3	186,7	Ya
70	70,0%	27,1%	44,5	126,8	191,3	Ya
80	80,0%	45,8%	51,4	140,3	206,1	Ya
85	85,0%	57,3%	59,1	157,4	227,4	Ya
90	90,0%	70,2%	75,1	197,0	283,5	Ya
95	95,0%	84,4%	124,4	333,5	494,6	Tidak

4.3 Trade-off Biaya

Biaya per tenant turun berbanding terbalik dengan N (Gambar 2). Pada nilai dasar N = 60 biayanya Rp73.000 per tenant per bulan, dan pada ambang N_a = 90 turun menjadi Rp48.667. Tabel 4 membandingkan konsolidasi pada ambang tersebut dengan isolasi penuh. Isolasi penuh menghasilkan p95 lebih rendah (130,2 ms) karena tiap tenant hanya memakai 8% kapasitas worker-nya, tetapi biayanya Rp547.500 per tenant per bulan. Konsolidasi pada ambang SLO menghemat 91,1% biaya dengan tetap memenuhi SLO.



Gambar 2. Biaya per tenant per bulan terhadap jumlah tenant per node (hanya titik yang memenuhi SLO p95 200 ms), dibandingkan dengan isolasi penuh.

Tabel 4. Perbandingan isolasi penuh dan konsolidasi pada ambang SLO

Skema	Tenant per node	Worker	Utilisasi	p95 (ms)	Biaya per tenant/bulan
Isolasi penuh	1	1	8,0%	130,2	Rp547.500
Konsolidasi pada N_a	90	8	90,0%	197,0	Rp48.667

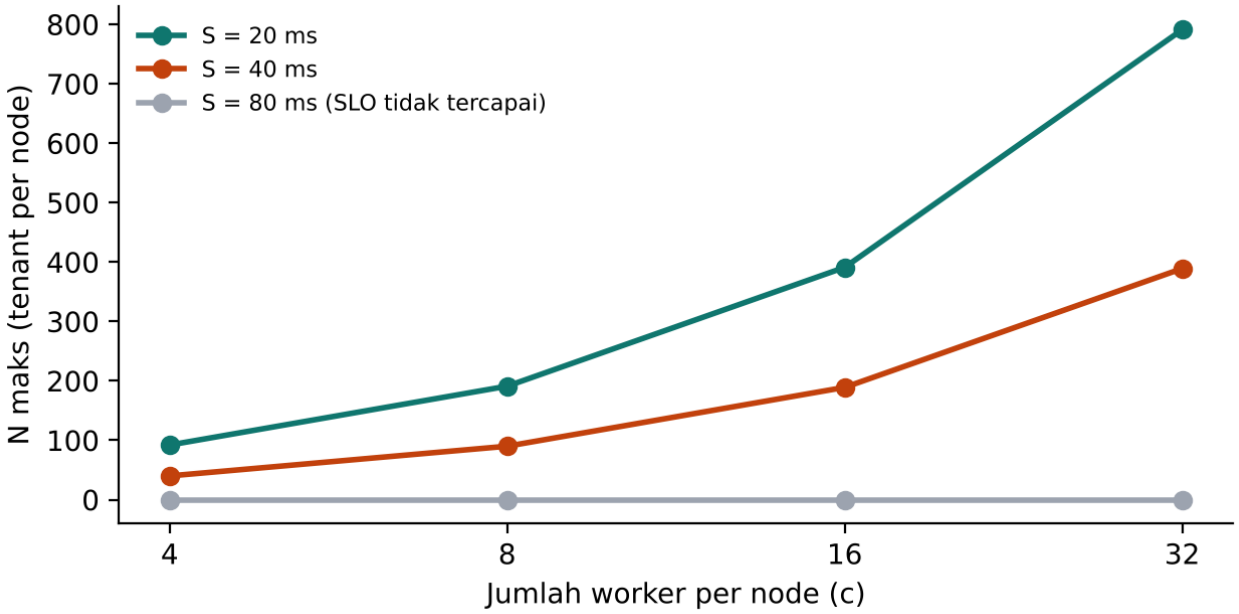
4.4 Analisis Sensitivitas

Tabel 5 dan Gambar 3 memperlihatkan N_a untuk kombinasi jumlah worker dan waktu layanan. Dengan S = 40 ms, menambah worker dari 4 ke 32 menaikkan N_a hampir sepuluh kali lipat (40 ke 389) dan menaikkan utilisasi yang aman dari 80,0% ke 97,3%, sehingga biaya per tenant pada ambang turun dari Rp54.750 ke Rp45.039. Sebaliknya, dengan S = 80 ms tidak ada konfigurasi yang memenuhi SLO.

Tabel 5. Ambang konsolidasi untuk berbagai jumlah worker dan waktu layanan (λ = 2 permintaan/detik, SLO p95 200 ms)

Worker (c)	Layanan (ms)	N_a	Utilisasi pada N_a	Biaya per tenant pada N_a
4	20	92	92,0%	Rp23.804
4	40	40	80,0%	Rp54.750
4	80	SLO tidak tercapai	–	–
8	20	191	95,5%	Rp22.932
8	40	90	90,0%	Rp48.667
8	80	SLO tidak tercapai	–	–
16	20	391	97,8%	Rp22.404
16	40	189	94,5%	Rp46.349
16	80	SLO tidak tercapai	–	–
32	20	791	98,9%	Rp22.149
32	40	389	97,3%	Rp45.039

Worker (c)	Layanan (ms)	N_a	Utilisasi pada N_a	Biaya per tenant pada N_a
32	80	SLO tidak tercapai	–	–



Gambar 3. Ambang konsolidasi N_a terhadap jumlah worker per node untuk tiga waktu layanan rata-rata.

5. Pembahasan

Temuan utama adalah adanya titik belok (knee) yang tegas: di bawah utilisasi sekitar 75% menambah tenant hampir tidak mengubah latensi, sedangkan di atas 80% setiap tambahan tenant memperpanjang ekor latensi secara cepat. Pola ini sejalan dengan argumen Dean dan Barroso (2013) bahwa ekor latensi memburuk lebih cepat daripada rata-ratanya saat utilisasi naik. Pada N = 90, latensi rata-rata baru 75,1 ms, jauh di bawah SLO, padahal p95 sudah 197,0 ms. Perencana kapasitas yang memakai rata-rata akan menyimpulkan bahwa node masih longgar, sehingga berisiko melanggar SLO.

Hasil sensitivitas mengonfirmasi efek skala pada server farm yang diuraikan Harchol-Balter (2013): node dengan lebih banyak worker dapat dijalankan pada utilisasi lebih tinggi untuk SLO yang sama. Bagi penyedia SaaS, ini berarti konsolidasi pada node besar lebih hemat per tenant daripada banyak node kecil, selama isolasi kinerja antar-tenant dijaga (Krebs dkk., 2012). Kasus S = 80 ms menunjukkan batas yang tidak bisa diatasi dengan menambah perangkat keras: pada layanan eksponensial, p95 waktu layanan saja sudah $-\ln(0{,}05) \times 80 \approx 239{,}7$ ms, melebihi SLO. Optimisasi kode atau pemecahan permintaan menjadi syarat sebelum konsolidasi.

Penelitian ini memiliki beberapa keterbatasan. Pertama, tenant diasumsikan homogen dengan kedatangan Poisson, padahal beban nyata sering bersifat ledakan (bursty) dan berbeda antar-tenant; Lorido-Botran dkk. (2014) mencatat pentingnya pola beban dalam penskalaan. Kedatangan bergelombang akan menurunkan N_a, sehingga hasil di sini sebaiknya dibaca sebagai batas atas. Kedua, waktu layanan eksponensial dapat meremehkan ekor pada beban berekor panjang, yang memerlukan model M/G/c seperti yang dibahas Khazaei dkk. (2012). Ketiga, harga diasumsikan linear terhadap jumlah worker dan model belum memasukkan gangguan antar-tenant (noisy neighbor) di tingkat basis data maupun memori. Keempat, model belum divalidasi dengan pengukuran sistem produksi. Lab Simulasi yang menyertai artikel memungkinkan pembaca mengganti parameter sesuai beban mereka dan menguji di titik mana klaim artikel mulai gagal.

6. Kesimpulan

Model antrean M/M/c dengan distribusi waktu tinggal eksak memberikan cara sederhana dan tervalidasi untuk menentukan ambang konsolidasi tenant SaaS terhadap SLO latensi p95. Pada konfigurasi dasar, ambangnya adalah 90 tenant per node, dengan penghematan biaya 91,1% dibanding isolasi penuh dan galat model terhadap simulasi di bawah 0,3%. Keputusan konsolidasi sebaiknya didasarkan pada persentil latensi, bukan rata-rata. Menambah worker per node menaikkan utilisasi yang aman, tetapi tidak dapat menolong layanan yang waktu layanannya sendiri sudah melampaui SLO. Penelitian lanjutan akan memasukkan kedatangan bergelombang, tenant heterogen, dan validasi dengan data operasional.

Pernyataan

Ketersediaan data dan kode. Kode (model_saas.py, validasi_des.py, analisis.py, gambar.py) dan data (sapuan_tenant.csv, validasi_des.csv, sensitivitas.csv, perbandingan_biaya.csv) tersedia sebagai berkas Kode & Data artikel ini. Seluruh data dihasilkan oleh model dan simulasi yang dijelaskan di Bagian 3; tidak ada data pengukuran sistem produksi.

Pernyataan penggunaan AI. Penyusunan draf naskah, kode, dan analisis dibantu asisten AI generatif (Claude, Anthropic). Penulis memeriksa seluruh isi, menjalankan ulang kode, dan bertanggung jawab penuh atas naskah ini.

Konflik kepentingan. Penulis menyatakan tidak ada konflik kepentingan.

Pendanaan. Penelitian ini tidak menerima pendanaan khusus.

Lampiran A. Klaim Utama Artikel

Tiga klaim berikut diterbitkan sebagai nanopublikasi bersama artikel ini:

1. Penambahan tenant per node SaaS multi-tenant meningkatkan latensi p95 secara nonlinier setelah utilisasi sekitar 80% (Gambar 1, Tabel 3).
2. Konsolidasi hingga ambang SLO menurunkan biaya infrastruktur per tenant sebesar 91,1% dibanding isolasi penuh (Gambar 2, Tabel 4).
3. Model M/M/c dengan distribusi waktu tinggal eksak menghasilkan prediksi latensi p95 dengan galat kurang dari 0,3% terhadap simulasi kejadian diskrit (Tabel 2).

Pernyataan Penggunaan AI

Penulis menggunakan Gemini Argon 4 untuk pembuatan kode program, analisis data, penyuntingan bahasa & tata bahasa. Seluruh hasil bantuan AI telah ditinjau dan disunting oleh penulis, yang bertanggung jawab penuh atas isi publikasi.

Daftar Pustaka

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50–58. https://doi.org/10.1145/1721654.1721672

Bezemer, C.-P., & Zaidman, A. (2010). Multi-tenant SaaS applications: Maintenance dream or nightmare? In Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) (pp. 88–92). ACM. https://doi.org/10.1145/1862372.1862393

Dean, J., & Barroso, L. A. (2013). The tail at scale. Communications of the ACM, 56(2), 74–80. https://doi.org/10.1145/2408776.2408794

Fehling, C., Leymann, F., & Mietzner, R. (2010). A framework for optimized distribution of tenants in cloud applications. In 2010 IEEE 3rd International Conference on Cloud Computing (pp. 252–259). IEEE.

Harchol-Balter, M. (2013). Performance modeling and design of computer systems: Queueing theory in action. Cambridge University Press.

Kabbedijk, J., Bezemer, C.-P., Jansen, S., & Zaidman, A. (2015). Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective. Journal of Systems and Software. https://doi.org/10.1016/j.jss.2014.10.034

Khazaei, H., Mišić, J., & Mišić, V. B. (2012). Performance analysis of cloud computing centers using M/G/m/m+r queuing systems. IEEE Transactions on Parallel and Distributed Systems, 23(5), 936–943. https://doi.org/10.1109/TPDS.2011.199

Krebs, R., Momm, C., & Kounev, S. (2012). Architectural concerns in multi-tenant SaaS applications. In Proceedings of the 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012) (pp. 426–431). SciTePress. https://doi.org/10.5220/0003957604260431

Lorido-Botran, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12(4), 559–592. https://doi.org/10.1007/s10723-014-9314-7

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (NIST Special Publication 800-145). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-145