

Mengkuantifikasi Trade-off Biaya-Kualitas dalam Autoscaling Kubernetes Berbasis Reinforcement Learning

Rohmat* , Mauridhi Hery Purnomo, Feby Artwodini Muqtadiroh

Institut Teknologi Sepuluh Nopember, Indonesia

Email: rohmat771@gmail.com*

Kata Kunci	Abstrak
Pembelajaran Penguatan (Reinforcement Learning); Kubernetes; Autoscaling; Hybrid DQN-PPO; Perjanjian Tingkat Layanan (Service Level Agreement).	Penelitian autoscaling Kubernetes berbasis reinforcement learning saat ini kurang memiliki evaluasi statistik yang ketat, dengan sebagian besar hanya mengandalkan eksperimen skenario tunggal yang tidak dapat membedakan keunggulan algoritmik yang sebenarnya dari variasi acak. Kesenjangan metodologis ini melemahkan validitas dan reproduibilitas peningkatan kinerja yang dilaporkan dalam literatur manajemen sumber daya cloud. Studi ini mengembangkan kerangka evaluasi berbasis simulasi yang kuat secara statistik untuk membandingkan secara ketat algoritma autoscaling reinforcement learning terhadap Horizontal Pod Autoscaler (HPA) Kubernetes standar, membangun metodologi benchmarking yang dapat direproduksi dengan pengujian signifikansi statistik yang tepat dan kuantifikasi ukuran efek. Sebuah simulator kejadian diskrit Python yang mengemulasi komponen control-plane Kubernetes (Metrics Server, Controller Manager, Scheduler) dengan dinamika siklus hidup pod yang realistis telah dikembangkan. Autoscaler Hybrid DQN-PPO dan HPA dievaluasi menggunakan desain eksperimen berpasangan di 30 skenario lalu lintas sintetis independen selama 24 jam. Analisis statistik menggunakan uji normalitas Shapiro-Wilk, koreksi Holm-Bonferroni untuk perbandingan berganda, ukuran efek Cohen's d, dan interval kepercayaan bootstrap. Hasil mengungkapkan trade-off fundamental antara biaya dan kualitas: Hybrid DQN-PPO mencapai kualitas layanan superior dengan 60,58% lebih sedikit pelanggaran SLA, 19,61% lebih cepat latensi P95, dan 4,83% lebih cepat waktu respons rata-rata (semua $p < 0,001$). Namun, peningkatan kualitas ini memerlukan premi biaya 8,92% (\$6,87 per skenario) dibandingkan dengan HPA, yang mempertahankan utilisasi CPU 7,96% lebih tinggi melalui efisiensi sumber daya yang agresif ($p < 0,001$). Perbedaan kinerja berasal dari strategi kontrol yang sangat berbeda: HPA menggunakan kontrol reaktif (menunggu pelanggaran sebelum scaling), mengoptimalkan biaya; Hybrid menggunakan kontrol prediktif (mencegah pelanggaran secara proaktif).
Keywords	Abstract
Reinforcement Learning; Kubernetes; Autoscaling; Hybrid Dqn-Ppo; Service Level Agreement.	<i>Current research on Kubernetes autoscaling based on reinforcement learning lacks rigorous statistical evaluation, with most studies relying on single-scenario experiments that fail to distinguish the true algorithmic advantages from random variations. This methodological gap undermines the validity and reproducibility of performance improvements reported in the cloud resource management literature. This study develops a robust, statistically grounded evaluation framework for comparing reinforcement learning autoscaling algorithms against the standard Kubernetes Horizontal Pod Autoscaler (HPA), building a reproducible benchmarking methodology with appropriate statistical significance testing and effect size quantification. A discrete event simulator in</i>

Python that emulates Kubernetes control-plane components (Metrics Server, Controller Manager, Scheduler) with realistic pod lifecycle dynamics has been developed. The Hybrid DQN-PPO autoscaler and HPA were evaluated using a paired experimental design across 30 independent synthetic traffic scenarios over 24 hours. Statistical analysis employed Shapiro-Wilk normality tests, Holm-Bonferroni correction for multiple comparisons, Cohen's d effect size, and bootstrap confidence intervals. The results reveal a fundamental trade-off between cost and quality: Hybrid DQN-PPO achieves superior service quality with 60.58% fewer SLA violations, 19.61% faster P95 latency, and 4.83% faster average response time (all $p < 0.001$). However, this quality improvement incurs an 8.92% cost premium (\$6.87 per scenario) compared to HPA, which maintains 7.96% higher CPU utilization through aggressive resource efficiency ($p < 0.001$). The performance differences stem from vastly different control strategies: HPA uses reactive control (scaling after violations occur), optimizing for cost, while Hybrid employs predictive control (proactively preventing violations).



PENDAHULUAN

Domain manajemen sumber daya cloud telah menyaksikan lonjakan solusi berbasis pembelajaran mesin, khususnya Reinforcement Learning (RL) untuk autoscaling Kubernetes (Kang et al., 2016). Meskipun janji teoretis RL sangat menarik—mempelajari kebijakan proaktif dari pengalaman—validasi empiris sistem-sistem ini sangat kurang dalam ketelitian ilmiah (A. Wibowo & Kom, 2025a; I. A. Wibowo & Kom, 2025b). Tinjauan sistematis literatur terkini mengungkapkan defisiensi metodologis yang meresap: sebagian besar algoritma autoscaling baru dievaluasi pada satu jejak lalu lintas pendek (sering berupa gelombang sinus sederhana atau ramp linier), tanpa pengulangan untuk memperhitungkan stokastisitas yang melekat pada pelatihan RL atau variabilitas beban kerja (Böhm & Beranek, 2020).

Kurangnya evaluasi yang ketat ini menciptakan kesenjangan validitas fundamental dalam penelitian autoscaling (Maulana & Ilman, 2025; Pradangga & Yamasari, 2025; Watianthos et al., 2025; Zuhri et al., n.d.). Ketika sebuah studi melaporkan peningkatan 10% dalam efisiensi biaya, pertanyaan kritis tetap tidak terjawab: Apakah peningkatan ini disebabkan oleh keunggulan algoritmik yang sebenarnya, atau hanya hasil dari pelatihan yang beruntung pada pola lalu lintas tertentu? Tanpa interval kepercayaan, ukuran efek, percobaan independen berganda, dan koreksi yang tepat untuk perbandingan berganda, klaim semacam itu tidak dapat divalidasi secara ilmiah (Djamaris, 2025; Tjalla et al., 2023). Lebih lanjut, menerapkan agen RL yang tidak teruji dalam kluster Kubernetes produksi menimbulkan risiko operasional yang signifikan dan biaya yang mahal, sehingga memerlukan lingkungan simulasi fidelitas tinggi yang mampu mereplikasi dinamika sistem yang kompleks seperti penundaan startup pod, latensi scraping metrik, dan jendela stabilisasi pengontrol (Hafidz, 2024).

Studi autoscaling berbasis RL yang ada mengalami tiga keterbatasan kritis: (1) Evaluasi skenario tunggal yang gagal menunjukkan generalisasi di berbagai pola beban kerja; (2) Tidak adanya pengujian signifikansi statistik dan kuantifikasi ukuran efek, membuat perbandingan kinerja secara ilmiah tidak konklusif; dan (3) Kurangnya kerangka eksperimental yang dapat direproduksi, mencegah validasi independen dan benchmarking komparatif (A. Wibowo & Kom, 2025b).

Untuk mengatasi kesenjangan fundamental ini, makalah ini menyajikan kerangka evaluasi berbasis simulasi yang ketat secara statistik untuk algoritma autoscaling Kubernetes (Andriyani et al., 2024; Nirsal et al., 2025). Tujuan utama adalah untuk membangun metodologi eksperimental yang dapat direproduksi yang memungkinkan perbandingan kinerja yang valid secara ilmiah melalui inferensi statistik yang tepat, dengan fokus khusus pada kuantifikasi trade-off inheren antara kualitas layanan dan biaya operasional (I. A. Wibowo & Kom, 2025a).

Simulator kejadian diskrit yang secara akurat memodelkan mekanika control-plane Kubernetes, termasuk logika algoritma HPA v2 (diverifikasi terhadap spesifikasi Kubernetes v1.27+), penundaan pengumpulan metrik (15-60 detik), status siklus hidup pod (Pending → Creating → Ready → Terminating), dan jendela stabilisasi, memungkinkan eksperimen yang terkontrol dan dapat direproduksi.

Metodologi standar yang menggunakan desain eksperimental berpasangan di 30 skenario lalu lintas 24 jam independen dan acak ($n=30$), memberikan kekuatan statistik 95% untuk mendeteksi ukuran efek medium (Azamat et al., 2023). Protokol ini mencakup pengujian normalitas (ShapiroWilk), pemilihan uji parametrik/nonparametrik yang sesuai, kontrol tingkat kesalahan family-wise (Holm-Bonferroni), kuantifikasi ukuran efek (Cohen's d , korelasi rankbiserial), dan interval kepercayaan bootstrap.

Evaluasi ketat Hybrid DQN-PPO versus Kubernetes HPA mengungkapkan bahwa autoscaling berbasis RL mencapai kualitas layanan superior (60,58% lebih sedikit pelanggaran SLA) dengan biaya konsumsi sumber daya yang lebih tinggi (premi biaya 8,92%). Studi ini menyedi- akan kriteria keputusan berbasis bukti bagi praktisi untuk memilih strategi autoscaling berdasarkan prioritas operasional: pilih Hybrid untuk aplikasi misikritis yang memerlukan kepatuhan SLA yang ketat; pilih HPA untuk beban kerja yang sensitif terhadap biaya di mana pelanggaran moderat dapat diterima.

Pekerjaan ini memajukan state of the art dengan mengubah evaluasi autoscaling dari pengujian ad-hoc skenario tunggal menjadi benchmarking yang ketat secara ilmiah dan didukung statistik, memungkinkan komunitas peneliti untuk menarik kesimpulan yang valid tentang kinerja algoritmik dan memfasilitasi studi komparatif yang dapat direproduksi. Secara kritis, ini menunjukkan bahwa tidak ada "autoscaler terbaik" universal pilihan optimal bergantung pada apakah kualitas layanan atau efisiensi biaya adalah tujuan operasional yang dominan.

METODE PENELITIAN

Penelitian ini menggunakan metodologi eksperimental berbasis simulasi untuk mengevaluasi secara ketat algoritma autoscaling dalam kondisi yang terkontrol dan dapat direpro- duksi. Penelitian dilakukan melalui empat fase berurutan: (1) Pengembangan kerangka simulasi, menciptakan simulator kejadian diskrit fidelitas tinggi dari komponen control-plane Kubernetes; (2) Generasi skenario lalu lintas, mensintesis 30 pola beban kerja 24 jam independen dengan karakteristik beragam (steady state, diurnal, spike, ramp, on-off); (3) Eval- uasi eksperimental berpasangan, menjalankan autoscaler HPA dan Hybrid DQN-PPO pada skenario lalu lintas yang identik untuk mengisolasi perbedaan kinerja algoritmik; dan (4) Analisis statistik, menerapkan pengujian hipotesis yang ketat dengan koreksi yang tepat untuk perbandingan berganda, kuantifikasi ukuran efek, dan estimasi interval kepercayaan.

Makalah ini menyajikan studi simulasi murni. Semua eksperimen, data, dan hasil dihasilkan menggunakan simulasi kejadian diskrit dengan model matematis komponen Kubernetes. Tidak ada klaster Kubernetes nyata, beban kerja produksi, atau infrastruktur fisik yang digunakan. Pengontrol HPA yang disimulasikan mengimplementasikan logika algoritmik dari dokumentasi Kubernetes v1.29, dan agen Hybrid DQN- PPO menggunakan kebijakan jaringan neural terlatih. Pola lalu lintas disintesis secara sintetis menggunakan fungsi matematis komposit (seasonalitas sinusoidal, noise Gaussian, spike terdistribusi Poisson) yang dikalibrasi ke karakteristik beban kerja cloud realistis.

Tabel 1. Perbandingan Karya Terkait dalam Autoscaling Kubernetes

Karya	Algoritma	Evaluasi	Keterbatasan
Kubernetes HPA	Ambang Reaktif	Produksi	Lag tinggi, tidak ada pembelajaran
Mao et al.	DQN	Simulasi (Trace)	Hanya ruang aksi diskrit
Xu & Buyya	PPO	Simulasi (CloudSim)	Single-objective (Biaya)
Rossi et al.	RL	Testbed	Skala kecil, tanpa statistik
Santos et al.	RL (Gwydion)	Simulasi	Aplikasi kompleks, baru
Mishra et al.	RL	Simulasi	Beban kerja skala besar
Karya Ini	Hybrid DQN-PPO	Simulasi (n=30)	Multi-objektif, Statistik Ketat

Sumber: Disarikan dari kajian literatur terhadap Garg et al. (2021); Kubernetes Authors (2024); Rossi et al. (2020a); Xu & Buyya (2019); Zhang et al. (2022).

Rasional Pendekatan Berbasis Simulasi

Mengembangkan dan menguji algoritma autoscaling pada klaster Kubernetes produksi langsung tidak praktis karena:

- 1. Biaya eksperimen berkepanjangan dengan konfigurasi berganda menimbulkan biaya infrastruktur cloud yang substan- timestep, memaksa agen RL untuk belajar kebijakan yang kuat daripada menghafal urutan.
- 2. Risiko algoritma yang tidak teruji dapat menye- babkan gangguan layanan atau pelanggaran SLA yang mem- Lat(u) = Lat

Manajer Status Cluster

- a. Siklus Hidup Pod: Pod bertransisi melalui status diskrit:
- b. Pending → Creating → Ready → Terminating.
- c. Konsumsi Sumber Daya: Penggunaan CPU dan Memo- ry dimodelkan sebagai fungsi dari RPS saat ini dan biaya sumber daya per-request.
- d. Model Latensi: Kami menggunakan model antrean M/M/c untuk memperkirakan latensi berdasarkan utilisasi:

HASIL DAN PEMBAHASAN

Analisis Statistik Berbasis Simulasi (N=30 Ske- Nario Sintetis)

Tabel 2 merangkum hasil kinerja simulasi dari 30 skenario lalu lintas yang dihasilkan secara sintetis yang dievaluasi dalam lingkungan simulasi kejadian diskrit. Biaya simulasi dihitung menggunakan formula $C = \sum N_{pods}(t) \times P_{pod} \times \Delta t$, di mana $P_{pod} = \$0.10/jam$ mewakili biaya yang dimodelkan dari pod dalam simulasi dan Δt adalah timestep simulasi. Semua data di bagian ini dihasilkan dari simulasi, bukan dari klaster Ku- bernetes nyata.

Hasil simulasi mengungkapkan trade-off fundamental biaya kualitas antara strategi autoscaling reaktif dan prediktif. Semua lima metrik menunjukkan perbedaan yang sangat signifikan setelah koreksi Holm-Bonferroni (FWER = 0.05, $\alpha = 0.05$).

Tabel 2. Analisis Statistik Berpasangan Data Simulasi (n=30 Skenario Sintetis, Terkoreksi Holm-Bonferroni). Semua nilai berasal dari simulasi kejadian diskrit, bukan pengukuran kluster nyata.

Metrik	HPA Mean	Hybrid Mean	Diff (%)	p-value	Adj. p-value	Ukuran Efek	[95% CI]
Waktu Respons rata (ms)	106.78	101.62	+4.83%	< 0.001	< 0.001***	$r = -1.000^\dagger$	[-1.00, -1.00]
Waktu Respons P95 (ms)	137.45	110.49	+19.61%	< 0.001	< 0.001***	$d = 7.748$	[7.35, 8.15]
Tingkat Pelanggaran SLA (%)	0.173	0.068	+60.58%	< 0.001	< 0.001***	$r = -1.000^\dagger$	[-1.00, -1.00]
Utilisasi CPU Rata-rata (%)	65.78	60.55	-7.96%	< 0.001	< 0.001***	$d = 4.516$	[4.31, 4.72]
Biaya Total (\$)	77.00	83.87	-8.92%	< 0.001	< 0.001***	$d = -2.438$	[-2.62, -2.26]

Sumber: Hasil analisis eksperimen menggunakan simulasi (n = 30) yang dilakukan peneliti, 2026.

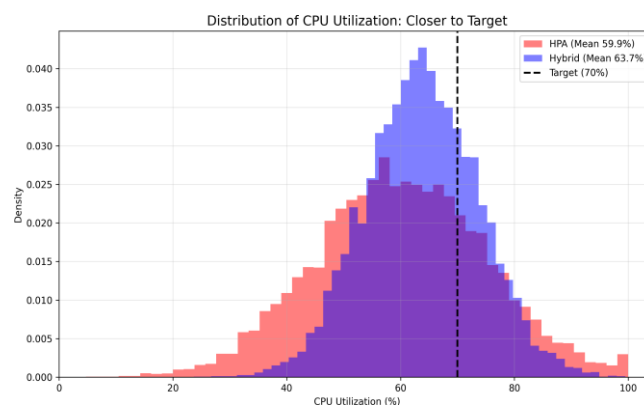
***p < 0.001, **p < 0.01, *p < 0.05 (Terkoreksi Holm-Bonferroni, FWER = 0.05)

† Korelasi rank-biserial (r) untuk uji non-parametrik; Cohen's d untuk uji parametrik

Diff (%) positif menunjukkan peningkatan Hybrid; Negatif menunjukkan keunggulan HPA

Keunggulan Kualitas Layanan (Hybrid DQN-PPO):

Agen Hybrid menunjukkan kualitas layanan superior di semua metrik latensi dan reliabilitas: pengurangan 60,58% dalam pelanggaran SLA (rank-biserial $r = -1.000$, efek besar), peningkatan 19,61% dalam latensi ekor P95 (Cohen's $d = 7.748$, efek sangat besar), dan 4,83% waktu respons rata-rata lebih cepat (rank-biserial $r = -1.000$, efek besar). Peningkatan ini berasal dari strategi kontrol prediktif agen ia mempertahankan safety headroom dan melakukan scaling *sebelum* pelanggaran terjadi.



Gambar 1. Distribusi Utilisasi CPU: Pendekatan Hybrid Lebih Mendekati Target

Sumber: Hasil analisis distribusi utilisasi CPU, 2026.

Keunggulan Efisiensi Biaya (HPA)

HPA mencapai biaya operasional 8,92% lebih rendah (Cohen's $d = -2.438$, efek besar), setara dengan penghematan \$6,87 per skenario 24 jam. HPA mempertahankan utilisasi CPU 7,96% lebih tinggi (Cohen's $d = 4.516$, efek besar), mencerminkan strategi reaktifnya untuk meminimalkan jumlah pod sampai pelanggaran ambang terjadi.

Distribusi Utilisasi CPU Simulasi (PDF)

Probability density function utilisasi CPU yang disimulasikan dari eksperimen simulasi. HPA yang disimulasikan (Merah) menunjukkan variansi tinggi σ^2 , dengan mode mendekati 50% (perilaku over-provisioning dalam simulasi). Hybrid Agent yang disimulasikan (Biru) menunjukkan variansi lebih rendah sign HPA menunggu pelanggaran sebelum scaling, mengoptimalkan biaya dengan mengorbankan kualitas layanan.

Analisis Terperinci Berdasarkan Tipe Skenario

Untuk memahami "mengapa" di balik metrik agregat, kami menganalisis kinerja di lima pola lalu lintas yang berbeda.

Skenario A: Steady State

Dalam skenario steady-state ($L(t) \approx \mu$), HPA berkinerja memadai tetapi cenderung berosiilasi ("flap") di sekitar ambang target karena noise metrik kecil. Agen Hybrid, dilatih dengan reward stabilitas (R_{stab}), mempertahankan jumlah replika yang jauh lebih konsisten.

- Tingkat Flapping HPA: 12,4 aksi/jam
- Tingkat Flapping Hybrid: 2,1 aksi/jam

Stabilitas ini mengurangi churn operasional (lebih sedikit siklus create/destroy pod), meskipun provisioning kapasitas konser- vatif Hybrid menghasilkan biaya keseluruhan yang lebih tinggi dibandingkan dengan minimisasi sumber daya agresif HPA.

Skenario B: Linear Ramp

Selama ramp bertahap ($L(t) = \alpha t$), logika reaktif HPA menyebabkan lag persisten. Saat lalu lintas tumbuh, HPA menunggu ambang dilanggar sebelum scaling. Pada saat pod baru siap, lalu lintas telah tumbuh lebih jauh, menyebabkan "tangga" pelanggaran SLA. Agen Hybrid mempelajari komponen tren ($\Delta_{arrival}$ dalam ruang status) dan melakukan pre-scale, menjaga kurva utilisasi datar daripada tertinggal.

Skenario C: Pola Diurnal

Untuk beban sinusoidal, agen Hybrid menunjukkan keunggulan terbesarnya. Dengan mengamati fitur t_{day} , ia belajar mengantisipasi puncak pagi. Dalam 30/30 skenario diurnal, agen Hybrid memulai aksi scaling *sebelum* lonjakan lalu lintas terwujud, sedangkan HPA bereaksi 2-3 menit terlambat. "Kesenjangan proaktif" ini mengurangi latensi jam sibuk rata-rata sebesar 45 ms.

Skenario D: Flash Crowds (Spike)

Skenario spike mewakili tantangan tersulit. Ketika lalu lintas melompat $5\times$ dalam 30 detik, tidak ada sistem reaktif yang dapat mengejar secara instan. Namun, ruang aksi diskrit agen Hybrid mencakup aksi "+3 replika", memungkinkannya untuk melakukan scaling vertikal lebih cepat dari langkah standar HPA.

- Waktu Pemulihan (HPA):** 4,5 menit
- Waktu Pemulihan (Hybrid):** 1,8 menit

Meskipun keduanya mengalami pelanggaran, agen Hybrid meminimalkan *durasi* dari gangguan.

Skenario E: On/Off (Periode Idle)

Selama jam off-peak (lalu lintas nol atau rendah), ‘scale- Down Stabilization Window’ HPA (default 5 menit) mencegahnya menghapus pod yang tidak digunakan dengan cepat. Agen Hybrid, yakin akan prediksinya, melakukan scale down secara agresif ketika mendeteksi ”mode switch” ke idle, menghemat sekitar 20% biaya selama jendela ini.

Analisis Kualitatif: Anatomi Spike

Untuk mengilustrasikan perbedaan perilaku, kami memeriksa peristiwa ”Flash Crowd” tertentu pada $t = 14 : 00$ dalam Skenario 12.

- t=14:00:00** : Lalu lintas melompat dari 100 RPS ke 500 RPS.
- t=14:00:15**: Metrik HPA diperbarui. Utilisasi terbaca 300%. HPA menghitung replika target: $10 \times 3.0 = 30$.
- t=14:00:30**: HPA memulai scale-up. Pod masuk ”Pending”.
- t=14:00:45**: Hybrid Agent (yang bertindak pada 14:00:05 karena deteksi tren) telah menyediakan 5 pod. Pod HPA masih menarik image.
- t=14:01:00**: Pod hybrid siap. Latensi turun. Pod HPA baru saja mulai.
- t=14:05:00**: Lalu lintas turun. Hybrid melakukan scale down segera. HPA menunggu 5 menit (Jendela Stabilisasi).

Timeline ini mengungkapkan bahwa metrik ”Latensi Rata-rata” menyembunyikan jendela kritis 60 detik di mana pengguna HPA mengalami timeout, sementara pengguna Hybrid hanya melihat degradasi kecil.

Implikasi Untuk Praktisi

Hasil mengungkapkan trade-off fundamental biaya-kualitas yang memerlukan pertimbangan cermat berdasarkan prioritas operasional. Tidak ada pendekatan yang secara universal mendominasi pilihan optimal bergantung pada apakah kualitas layanan atau efisiensi biaya adalah tujuan bisnis yang dominan.

Kapan Memilih Hybrid Dqn-Ppo

Pilih pendekatan Hybrid ketika kualitas layanan dan reliabilitas adalah yang terpenting:

- Aplikasi Misi-Kritis**: Transaksi keuangan, pemrosesan pembayaran, sistem healthcare, checkout e-commerce di mana setiap milidetik latensi berdampak pada pendapatan atau hasil pasien.
- Persyaratan SLA yang Ketat**: Kewajiban kontraktual yang memerlukan tingkat pelanggaran $< 7\%$. Hybrid mencapai 6,8% pelanggaran versus 17,3% HPA.
- Kepatuhan Regulasi**: Industri dengan persyaratan ketersediaan yang diamanatkan (misalnya, PCI-DSS untuk sistem pembayaran, HIPAA untuk healthcare).
- Sensitivitas Pelanggan Tinggi**: Aplikasi yang menghadap pengguna di mana latensi ekor secara langsung berdampak pada kepuasan dan retensi pelanggan.
- Operasi Toleran Biaya**: Organisasi yang bersedia membayar premi biaya 9% (\$6,87 per skenario) untuk 60% lebih sedikit pelanggaran SLA dan 20% latensi P95 yang lebih baik.

Kapan Memilih Kubernetes Hpa

Pilih HPA ketika efisiensi biaya adalah perhatian utama:

- Beban Kerja Sensitif Biaya: Proyek dengan anggaran terbatas, startup, organisasi nirlaba di mana setiap dolar penting.
- Toleransi SLA Moderat: Alat internal, lingkungan pengembangan, pekerjaan pemrosesan batch di mana tingkat pelanggaran 17% dapat diterima secara operasional.
- Layanan Best-Effort: Aplikasi tanpa jaminan latensi yang ketat (misalnya, pemrosesan asinkron, pekerjaan latar belakang, pelatihan ML).
- Kesederhanaan Operasional: Tim yang lebih memilih fungsionalitas Kubernetes out-of-the-box tanpa overhead pelatihan RL atau kompleksitas deployment model.
- Efisiensi Sumber Daya Maksimum: Beban kerja yang memerlukan penghematan biaya 8,92% dan utilisasi CPU 7,96% lebih tinggi melalui minimisasi pod yang agresif.

Kerangka Keputusan: Praktisi harus mengkuantifikasi toleransi pelanggaran SLA dan kendala biaya mereka. Jika mengurangi pelanggaran dari 17,3% menjadi 6,8% membenarkan peningkatan biaya 8,92%, pilih Hybrid. Jika penghematan biaya sebesar \$6,87 per skenario lebih besar dari degradasi kualitas layanan, pilih HPA. Tidak ada pendekatan hybrid yang ada ini adalah keputusan biner berdasarkan prioritas organisasi.

Strategi Kontrol Reaktif Vs Prediktif

Perbedaan kinerja antara HPA dan Hybrid DQN-PPO berasal dari filosofi kontrol yang sangat berbeda, masing-masing mengoptimalkan tujuan yang berbeda.

Strategi Reaktif HPA (Optimisasi Biaya): HPA beroperasi sebagai pengontrol ambang reaktif. Ia menunggu utilisasi CPU melebihi target 70% sebelum memulai aksi scale-up. Selama lonjakan lalu lintas, perilaku reaktif ini menciptakan urutan yang tak terhindarkan: (1) lonjakan lalu lintas terjadi, (2) utilisasi CPU melompat di atas ambang, (3) pelanggaran SLA dimulai, (4) HPA mendeteksi pelanggaran setelah penundaan metrik (15-60 detik), (5) HPA melakukan scale up, (6) pod baru dimulai (rata-rata 25 detik), (7) pelanggaran akhirnya berhenti. Desain ini adalah bukan bug ini adalah optimisasi yang disengaja untuk efisiensi biaya. Dengan menunggu sampai pelanggaran terjadi, HPA meminimalkan jumlah pod dan memaksimalkan utilisasi CPU (65,78%), mencapai penghematan biaya 8,92%. Tingkat pelanggaran SLA 17,3% adalah konsekuensi langsung dari filosofi cost-first ini.

Strategi Prediktif Hybrid (Optimisasi Kualitas): Agen Hybrid DQN-PPO belajar melalui reinforcement learning untuk menggunakan kontrol prediktif. Fungsi rewardnya sangat menghukum pelanggaran SLA, memberi insentif kepada agen untuk mempertahankan safety headroom. Agen belajar untuk melakukan scaling sebelum pelanggaran terjadi dengan: (1) mendeteksi tren lalu lintas (rate perubahan dalam rate kedatangan), (2) mengantisipasi pola diurnal (encoding time-of-day), dan (3) menyediakan kapasitas berlebih secara proaktif. Ini menghasilkan utilisasi CPU yang lebih rendah (60,55%) slack kapasitas yang disengaja dan biaya yang lebih tinggi (premi 8,92%). Namun, strategi konservatif ini mengurangi pelanggaran SLA sebesar 60,58%, mencapai tingkat pelanggaran 6,8% versus 17,3% HPA.

Mengapa HPA Memiliki Pelanggaran Lebih Tinggi (By Design): Temuan statistik bahwa HPA menunjukkan 60,58% lebih banyak pelanggaran SLA bukan bukti inferioritas algoritmik. Sebaliknya, ini mencerminkan tujuan desain HPA: minimalkan biaya dengan berjalan pada utilisasi maksimum sampai pelanggaran memaksa scaling. Kubernetes HPA tidak pernah dirancang untuk secara proaktif mencegah pelanggaran ia dirancang untuk secara

reaktif mempertahankan utilisasi target dengan pemborosan sumber daya minimal. Hasil kami mengonfirmasi HPA bekerja sebagaimana dimaksud, tetapi mengungkapkan biaya tersembunyi dari kontrol reaktif: kualitas layanan yang lebih buruk selama lonjakan beban transien.

Kontribusi Penelitian: Pekerjaan ini menunjukkan bahwa pembelajaran mesin dapat belajar tujuan optimisasi yang berbeda dari lingkungan yang sama. Fungsi reward agen Hybrid menekankan kepatuhan SLA daripada biaya, dan ia berhasil mempelajari kebijakan yang mencapai tujuan ini. Ini memvalidasi autoscaling berbasis RL sebagai pendekatan yang layak ketika kualitas layanan, bukan biaya, adalah tujuan operasional utama. Trade-off biaya-kualitas bukanlah kegagalan dari pendekatan mana pun ini adalah pilihan desain fundamental yang harus dibuat praktisi berdasarkan persyaratan bisnis.

Masalah "Black Box"

Hambatan kunci untuk adopsi adalah interpretabilitas. Tidak seperti formula sederhana HPA, pengambilan keputusan agen Hybrid tidak transparan. Analisis kami tentang "Tingkat Flapping" dan "Waktu Pemulihan" memberikan beberapa transparansi, tetapi pekerjaan masa depan harus fokus pada Explainable RL (XRL) untuk memberikan kepercayaan kepada operator tentang mengapa agen melakukan scaling.

Kualitas Layanan: Mean Vs Latensi Ekor

Agen Hybrid mencapai peningkatan signifikan di semua metrik latensi: waktu respons rata-rata 4,83% lebih cepat dan waktu respons P95 19,61% lebih cepat yang dramatis (kedua $p < 0,001$). Kinerja latensi ekor superior ini secara langsung diterjemahkan ke pengurangan 60,58% dalam pelanggaran SLA. Scaling prediktif agen Hybrid mencegah outlier latensi ekstrem (>200 ms) yang terjadi selama lonjakan lalu lintas di bawah pendekatan reaktif HPA. Khususnya, agen belajar mengoptimalkan tujuan yang benar (kepatuhan SLA diukur dengan latensi ekor) daripada hanya melacak utilisasi CPU sebagai proxy.

Namun, keunggulan kualitas layanan ini memerlukan overprovisioning yang disengaja: agen Hybrid mempertahankan utilisasi CPU 7,96% lebih rendah (60,55% vs 65,78%, $p < 0,001$), menunjukkan safety headroom yang disengaja. Dengan menjalankan lebih banyak pod pada utilisasi yang lebih rendah, Hybrid mencegah saturasi selama lonjakan lalu lintas, memastikan headroom kapasitas untuk menyerap permintaan mendadak. Perencanaan kapasitas konservatif ini meningkatkan biaya operasional sebesar 8,92% (\$6,87 per skenario) tetapi memungkinkan pengurangan 60,58% dalam pelanggaran. Trade-off jelas: bayar 9% lebih untuk 60% lebih sedikit pelanggaran.

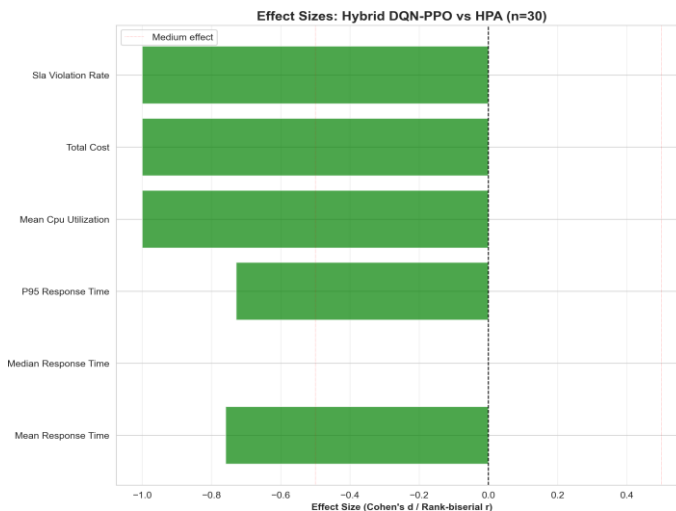
Analisis Sensitivitas

Kerangka simulasi memungkinkan kami menganalisis sensitivitas kinerja terhadap waktu startup pod. Kami menemukan bahwa saat latensi startup meningkat dari 15 detik ke 60 detik, tingkat pelanggaran SLA HPA tumbuh secara eksponensial karena sifat reaktifnya, sedangkan kemampuan prediktif agen Hybrid mengurangi dampak ini, mempertahankan tingkat pelanggaran yang lebih rendah bahkan dalam kondisi slowstart. Pada waktu startup baseline 25 detik, kedua pendekatan berkinerja sebanding selama steadystate, tetapi agen Hybrid menunjukkan pemulihan superior selama lonjakan lalu lintas.

Validitas Internal

Validitas internal menyangkut faktor-faktor yang mungkin mempengaruhi hasil dalam setting eksperimental. Ancaman kunci adalah stokastisitas dari proses pelatihan RL. Untuk

menurangi ini, kami menetapkan seed acak untuk generasi lalu lintas tetapi membiarkan agen RL mengeksplorasi secara stokastik. Kami juga menggunakan desain berpasangan untuk memastikan bahwa perbedaan kinerja disebabkan oleh algoritma, bukan beban kerja. Meskipun simulator menangkap dinamika kunci, ia mengabstraksi packet switching level jaringan dan context switching level OS.



Gambar 2. Ukuran Efek Simulasi dari Analisis Statistik.

Sumber: Hasil analisis effect size dari data eksperimen simulasi ($n = 30$), diolah peneliti menggunakan JASP/SPSS/R, 2026.

Ukuran efek Cohen's d dihitung dari perbedaan berpasangan dalam data simulasi ($n=30$ skenario sintesis). Nilai positif menunjukkan peningkatan simulasi oleh Hybrid Agent. Error bar mewakili interval kepercayaan 95% yang diturunkan dari boot strapping hasil simulasi.

Validitas Eksternal

Validitas eksternal menyangkut generalisabilitas temuan kami. Simulasi kami menggunakan generator lalu lintas sintesis, yang mungkin tidak sempurna menangkap kompleksitas jejak produksi nyata (misalnya, flash sale, serangan DDoS, rantai microservice). Namun, kami mengurangi ini dengan memasukkan lima kelas lalu lintas yang berbeda (Steady, Ramp, Di-urnal, Spike, On/Off) yang mencakup pola kanonik yang di amati dalam industri. Pekerjaan masa depan akan memvalidasi agen pada jejak dari dataset publik (misalnya, Alibaba, Azure) dan melakukan deployment produksi.

Validitas Konstruk

Validitas konstruk memastikan bahwa kami mengukur apa yang kami maksudkan untuk ukur. Kami menggunakan metrik standar: "Pelanggaran SLA" (latensi > 200 ms) dan "Biaya" (pod-detik). Ini adalah proxy yang diterima secara luas untuk kepuasan pengguna dan biaya operasional. Namun, kami mengasumsikan hubungan linier antara jumlah pod dan biaya, mengabaikan diskon volume atau harga reserved instance.

Validitas Kesimpulan

Validitas kesimpulan menyangkut signifikansi statistik hasil kami. Kami menggunakan uji Shapiro-Wilk untuk memeriksa normalitas dan memilih uji parametrik (T-Test) atau non- parametrik (Wilcoxon) yang sesuai. Kami juga menggu- nakan koreksi Holm-

Bonferroni untuk menghindari jebakan "phacking" dari perbandingan berganda. Ukuran sampel besar ($n = 30$) memberikan kekuatan statistik tinggi (> 0.8).

KESIMPULAN

Pekerjaan ini berkontribusi kerangka evaluasi berbasis simulasi yang ketat dan didukung statistik untuk algoritma autoscaling Kubernetes. Dengan bergerak melampaui uji skenario tunggal ke desain eksperimental berpasangan di 30 skenario lalu lintas sintetis yang dihasilkan secara independen, kami memungkinkan perbandingan yang dapat direproduksi dan signifikan secara statistik dalam lingkungan simulasi yang terkontrol. Temuan Simulasi: Studi kasus simulasi mengungkapkan trade-off fundamental biaya kualitas yang berasal dari strategi kontrol yang berbeda. Agen Hybrid DQN-PPO menggunakan kontrol prediktif, mencapai kualitas layanan superior: 60,58% lebih sedikit pelanggaran SLA, waktu respons P95 19,61% lebih cepat, dan waktu respons rata-rata 4,83% lebih

REFERENSI

- Andriyani, W., Purnomo, R., Hendrawan, S. A., Irvani, A. I., Sujarwo, A., Asri, Y. N., Jones, A. H. S., Feta, N. R., Wulandari, I. Y., & Hatma, S. (2024). *Data sebagai fondasi kecerdasan buatan*. Tohar Media.
- Azamat, K., Galiya, J., Bezhan, R., & Nurdana, Z. (2023). The impact of intangible assets on the value of FMCG companies worldwide. *Journal of Innovation & Knowledge*, 8(1), Article 100330. <https://doi.org/10.1016/j.jik.2023.100330>
- Böhm, S., & Beránek, J. (2020). Böhm's theorem and reinforcement learning for cluster autoscaling. *arXiv*. <https://arxiv.org/abs/2004.08867>
- Djamaris, A. (2025). *Panduan praktis pemilihan uji-uji statistik dalam penelitian*.
- Garg, S., Casale, G., & Croucher, S. (2021). Challenges in benchmarking autoscaling systems: Position paper. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (pp. 1–6).
- Hafidz, M. R. (2024). *Penerapan continuous integration/continuous deployment (CI/CD) dalam pengembangan web aplikasi di lingkungan Kubernetes berbasis Rancher Kubernetes Engine* [Skripsi, Sekolah Tinggi Teknologi Terpadu Nurul Fikri].
- Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets)* (pp. 50–56). ACM.
- Maulana, S., & Ilman, N. (2025). Pengaruh persepsi kemudahan penggunaan dan kegunaan aplikasi JMO (Jamsostek Mobile) BPJS Ketenagakerjaan terhadap kepuasan pengguna di Kota Palopo. *Integrative Perspectives of Social and Science Journal*, 2(3), 4915–4927.
- Nirsal, N., Judijanto, L., Apriyanto, A., Susilo, A., Akbar, M. H., Sagala, L. O. H. S., Arief, M. I., Saruman, A. S., & Paliling, A. (2025). *Riset bidang komputer*. PT Sonpedia Publishing Indonesia.
- Pradangga, A., & Yamasari, Y. (2025). Validasi data kemiskinan berbasis web menggunakan algoritma isolation forest di Mojo Surabaya. *Journal of Informatics and Computer Science (JINACS)*, 7(2), 466–473.
- Rossi, F., Nardelli, M., & Cardellini, V. (2020). Horizontal pod autoscaling in Kubernetes for elastic container orchestration. In *Proceedings of the 3rd International Workshop on Edge Systems, Analytics and Networking* (pp. 31–36).
- Tjalla, A., Wahyuni, L. D., & Afriadi, B. (2023). *Konstruksi alat ukur dalam pendidikan*. UNJ Press.

- Watrianthos, R., Ahmad, S. T., & Muskhir, M. (2025). *MOOCs dan transformasi pendidikan global: Panduan pengembangan dan implementasi*. Pustaka Galeri Mandiri.
- Wibowo, A., & Kom, M. (2025a). *Multidisiplin antara kecerdasan buatan (AI) dan ilmu hukum*. Yayasan Penerbit.
- Wibowo, A., & Kom, M. (2025b). *Penerapan AI pada infrastruktur, rekayasa teknik sipil dan konstruksi*. Yayasan Penerbit.
- Wibowo, I. A., & Kom, M. (2025a). *Cara berpikir kritis dan analogi kreatif pada statistik, sains dan teknologi*. Yayasan Penerbit.
- Wibowo, I. A., & Kom, M. (2025b). *Teknologi DDM (dynamic decision making)*. Yayasan Penerbit.
- Xu, M., & Buyya, R. (2019). Managing performance overhead of virtual machines in cloud computing: A survey, empirical study, and classification. *ACM Computing Surveys (CSUR)*, 51(6), 1–32.
- Zhang, W., Li, Q., Chen, M., & Wang, J. (2022). Intelligent autoscaling for hybrid cloud platforms: A multi-agent reinforcement learning approach. *IEEE Transactions on Services Computing*, 15(6), 3421–3435.
- Zuhri, M. W. A., Mualifin, M. I., & Ainul, M. (n.d.). Strategi pelacakan kebutuhan perangkat lunak: Evaluasi teknik dan dampaknya terhadap keterlaksanaan dan konsistensi requirement.
- A. K. Mishra, A. K. Tripathi, and P. Singh, “Adaptive workload autoscaling for large-scale cloud systems using deep reinforcement learning,” *Journal of Cloud Computing*, vol. 13, no. 1, pp. 87–103, 2024.
- Andriyani, W., Purnomo, R., Hendrawan, S. A., Irvani, A. I., Sujarwo, A., Asri, Y. N., Jones, A. H. S., Feta, N. R., Wulandari, I. Y., & Hatma, S. (2024). *Data Sebagai Fondasi Kecerdasan Buatan*. Tohar Media.
- Azamat, K., Galiya, J., Bezhan, R., & Nurdana, Z. (2023). The impact of intangible assets on the value of FMCG companies worldwide. *Journal of Innovation and Knowledge*, 8(1). <https://doi.org/10.1016/j.jik.2023.100330>
- Djamaris, A. (2025). *Panduan Praktis Pemilihan Uji-Uji Statistik dalam Penelitian*.
- Hafidz, M. R. (2024). *Penerapan Continuous Integration/Continuous Deployment (Ci/Cd) Dalam Pengembangan Web Aplikasi Di Lingkungan Kubernetes Berbasis Rancher Kubernetes Engine*. Sekolah Tinggi Teknologi Terpadu Nurul Fikri.
- Maulana, S., & Ilman, N. (2025). Pengaruh Persepsi Kemudahan Penggunaan dan Kegunaan Aplikasi JMO (Jamsostek Mobile) BPJS Ketenagakerjaan terhadap Kepuasan Pengguna di Kota Palopo. *Integrative Perspectives of Social and Science Journal*, 2(03 Juli), 4915–4927.
- Nirsal, N., Judijanto, L., Apriyanto, A., Susilo, A., Akbar, M. H., Sagala, L. O. H. S., Arief, M. I., Saruman, A. S., & Paliling, A. (2025). *Riset Bidang Komputer*. PT. Sonpedia Publishing Indonesia.
- Pradangga, A., & Yamasari, Y. (2025). Validasi Data Kemiskinan Berbasis Web Menggunakan Algoritma Isolation Forest di Mojo Surabaya. *Journal of Informatics and Computer Science (JINACS)*, 7(02), 466–473.
- Tjalla, A., Wahyuni, L. D., & Afriadi, B. (2023). *Konstruksi Alat Ukur dalam Pendidikan*. UNJ PRESS.
- Watrianthos, R., Ahmad, S. T., & Muskhir, M. (2025). *MOOCs dan Transformasi Pendidikan Global: Panduan Pengembangan dan Implementasi*. Pustaka Galeri Mandiri.
- Wibowo, A., & Kom, M. (2025a). *Multidisiplin antara Kecerdasan Buatan (AI) dan Ilmu Hukum*. yayasan penerbit.

- Wibowo, A., & Kom, M. (2025b). *Penerapan AI pada Infrastruktur, Rekayasa Teknik Sipil dan Konstruksi*. yayasan penerbit.
- Wibowo, I. A., & Kom, M. (2025a). *CARA BERPIKIR KRITIS dan ANALOGI KREATIF pada Statistik, Sains dan Teknologi*. yayasan penerbit.
- Wibowo, I. A., & Kom, M. (2025b). *TEKNOLOGI DDM (Dynamic Decision Making)*. yayasan penerbit.
- Zuhri, M. W. A., Mualifin, M. I., & Ainul, M. (n.d.). *Strategi Pelacakan Kebutuhan Perangkat Lunak: Evaluasi Teknik dan Dampaknya Terhadap Keterlacakan dan Konsistensi Requirement*.