ReTEC

Revista Electrónica de
Tecnología, Educación y Ciencia
ISSN: 2953-5654
http://retec.unsa.edu.ar
Universidad Nacional de Salta

# Eficiencia energética en autoescalado predictivo de Kubernetes: evaluación experimental con Kepler y RAPL

**Daniel Anunziata, Emilio Corti, Mercedes Carnero, José Luis Hernández**

Laboratorio de Redes · Grupo de Optimización · GCID (Grupo de Ciencia de Datos)
Facultad de Ingeniería · Universidad Nacional de Río Cuarto
{ danunziata, emiliocorti, jlh }@ing.unrc.edu.ar

**Revista Electrónica de Tecnología, Educación y Ciencia,**

**Volumen 1, Número 3, pág. 60-70, jun, 2026. ISSN: 2953-5654**

**Disponible en http://retec.unsa.edu.ar/**

# Eficiencia energética en autoescalado predictivo de Kubernetes: evaluación experimental con Kepler y RAPL

Daniel Anunziata, Emilio Corti, Mercedes Carnero, José Luis Hernández

Laboratorio de Redes · Grupo de Optimización · GCID (Grupo de Ciencia de Datos)
Facultad de Ingeniería · Universidad Nacional de Río Cuarto
{ danunziata, emiliocorti, jlh }@ing.unrc.edu.ar

**Resumen:** Los centros de datos consumen una proporción relevante de la electricidad mundial y su demanda crece con la expansión de servicios en la nube y cargas de inteligencia artificial. En ese contexto, las decisiones de autoescalado en Kubernetes no solo afectan la latencia y la disponibilidad, sino también la energía consumida por cada solicitud atendida. Este trabajo evalúa cinco estrategias de escalado —aprovisionamiento fijo, HPA al 50% de CPU, HPA al 70% de CPU, escalado predictivo basado en Long Short-Term Memory (LSTM) e híbrido predictivo-reactivo— midiendo julios por solicitud (J/req) con Kepler y contadores Running Average Power Limit (RAPL). Además, se contrastan esas mediciones con latencia P95, violaciones de nivel de servicio y pod-seconds. Sobre 81 experimentos válidos, analizados con pruebas no paramétricas, la estrategia predictiva reduce el consumo un 14,7% respecto del aprovisionamiento fijo y disminuye un 49% el uso de instancias. La estrategia híbrida aparece como la alternativa más conveniente para producción: con apenas 2,2% más de energía que el escalador reactivo, reduce las violaciones de nivel de servicio de 40,47% a 9,95%.

**Abstract:** Data centers account for a relevant share of global electricity consumption, and demand keeps increasing as cloud services and artificial intelligence workloads expand. In this context, Kubernetes autoscaling decisions affect not only latency and availability, but also the energy required to serve each request. This paper evaluates five scaling strategies —fixed provisioning, HPA at 50% CPU, HPA at 70% CPU, Long Short-Term Memory (LSTM)-based predictive scaling, and a predictive-reactive hybrid strategy— measuring joules per request (J/req) with Kepler and Running Average Power Limit (RAPL) counters. These measurements are analyzed together with P95 latency, service-level violations, and pod-seconds. Based on 81 valid experiments and a non-parametric statistical protocol, the predictive strategy reduces energy consumption by 14.7% compared with fixed provisioning and cuts pod usage by 49%. The hybrid strategy is the most suitable option for production: with only 2.2% additional energy compared with the reactive scaler, it reduces service-level violations from 40.47% to 9.95%.

**Palabras claves:** eficiencia energética; autoescalado predictivo; Kubernetes; Kepler; RAPL; LSTM; pruebas no paramétricas.

## 1. Introducción

La eficiencia energética se ha convertido en un criterio central para diseñar y operar plataformas en la nube. La razón es doble: por un lado, los centros de datos concentran una demanda eléctrica creciente; por otro, la expansión de servicios basados en inteligencia artificial aumenta la presión sobre la infraestructura computacional. En este escenario, una política de escalado ya

no puede evaluarse solo por su capacidad de responder rápido, sino también por la energía que consume para sostener esa respuesta.

Kubernetes resuelve parte de este problema mediante mecanismos de autoescalado. El enfoque más habitual es reactivo: el Horizontal Pod Autoscaler (HPA) ajusta la cantidad de réplicas cuando una métrica, como la utilización de CPU, supera un umbral. Esta decisión es simple y robusta, pero llega después de que la carga cambió. Durante ese intervalo puede aparecer subaprovisionamiento, con aumento de latencia y errores, o sobreaprovisionamiento, con consumo innecesario de recursos.

El autoescalado predictivo busca anticipar esos cambios. En lugar de esperar a que la CPU se sature, estima la demanda futura a partir de series temporales y prepara las réplicas antes de que ocurra el pico. En este trabajo se utiliza una red LSTM, adecuada para capturar dependencias temporales y patrones de carga recurrentes.

La pregunta que guía el artículo es concreta: ¿cuánto ahorro energético puede obtenerse con un escalador predictivo en Kubernetes y qué costo tiene ese ahorro en términos de calidad de servicio? Para responderla se comparan cinco estrategias bajo un mismo banco experimental, midiendo energía por solicitud, latencia P95, violaciones de nivel de servicio y uso efectivo de instancias.

La contribución principal es la cuantificación conjunta de eficiencia energética y disponibilidad. Los resultados muestran que el escalado predictivo puro minimiza el consumo, mientras que una estrategia híbrida, que combina predicción con una guarda reactiva de CPU, logra un equilibrio más adecuado para producción: protege los acuerdos de nivel de servicio con un sobrecosto energético muy bajo.

## 2. Trabajos relacionados

### 2.1. Medición de energía en contenedores

Medir energía en un entorno contenerizado no es trivial. Los contenedores comparten el mismo hardware físico y, por lo tanto, el consumo real se registra a nivel de procesador, memoria o nodo, no directamente a nivel de contenedor. RAPL proporciona contadores de energía para el paquete de CPU y, en algunas arquitecturas Intel, para el dominio DRAM. Kepler utiliza sondas eBPF para observar la actividad de los procesos y atribuir el consumo a los contenedores de manera proporcional al uso de CPU.

Esta atribución es especialmente útil para Kubernetes porque permite relacionar decisiones de escalado con energía consumida por cada carga de trabajo. Sin embargo, también tiene límites. En la plataforma experimental utilizada, basada en procesadores AMD Ryzen, no se dispone de medición RAPL para DRAM; por ello, la métrica energética reportada corresponde al dominio del paquete de CPU. Esta aclaración es relevante porque la carga de inferencia evaluada consume aproximadamente 1 GiB de memoria por instancia.

### 2.2. Autoescalado predictivo y eficiencia energética

La literatura sobre gestión de recursos en la nube ha abordado la eficiencia energética mediante consolidación de máquinas virtuales, modelos probabilísticos y políticas de asignación dinámica. En paralelo, los trabajos sobre autoescalado predictivo suelen concentrarse en latencia, tasa de errores o cumplimiento de SLA, pero no siempre miden energía en forma directa.

Ese desacople deja una brecha práctica: una estrategia puede reducir violaciones de servicio y, al mismo tiempo, consumir más energía de la necesaria; también puede ahorrar recursos, pero degradar la experiencia del usuario. El presente trabajo integra ambas dimensiones y evalúa el autoescalado predictivo no solo como una técnica de rendimiento, sino como una decisión energética.

## 3. Sistema de medición energética

### 3.1. Arquitectura de observabilidad

La arquitectura de observabilidad combina Prometheus 2.47, Kepler 0.7 y Grafana 10. Kepler se despliega como DaemonSet en cada nodo del clúster y exporta métricas energéticas a Prometheus cada 15 segundos. Grafana se utiliza para inspección y seguimiento visual de las series temporales.

La energía total de cada experimento se obtiene integrando la potencia instantánea observada durante el intervalo de medición:

$$E\_total = \Sigma\_i \, P\_i \cdot \Delta t$$

A partir de esa energía total se calcula la métrica central del estudio: julios por solicitud. Esta normalización permite comparar estrategias aunque procesen distintos volúmenes de carga:

$$J/req = E\_total \, / \, N\_requests$$

### 3.2. Estrategias evaluadas

Se evaluaron cinco estrategias de autoescalado. La Tabla 1 resume su principio de decisión y permite distinguir tres familias: aprovisionamiento fijo, escalado reactivo y escalado predictivo o híbrido.

**Tabla 1:** Estrategias de autoescalado evaluadas.

| Estrategia | Componente predictivo | Componente reactivo | Decisión |
|---|---|---|---|
| *fixed* | No | No (estático) | 24 instancias (C.2) / 16 instancias (C.4) |
| *hpa50* | No | CPU > 50% | HPA nativo vía KEDA |
| *hpa70* | No | CPU > 70% | HPA nativo vía KEDA (solo C.4) |
| *predictive* | LSTM MultiStep | No | Réplicas según predicción |
| *hybrid* | LSTM MultiStep | CPU guard | Máximo entre predicción y guarda reactiva |

La estrategia predictiva utiliza un modelo LSTM MultiStep de dos capas, con 64 y 32 unidades y Dropout de 0,2. El modelo fue entrenado sobre el conjunto sintético synthetic_v5, compuesto por 345.600 muestras equivalentes a 60 días de demanda, con un rango de 32,6 a 293,9 solicitudes por segundo. En evaluación alcanzó $R^2 = 0{,}924$ y MAPE = 7,38%.

La integración con Kubernetes se realiza mediante KEDA y su interfaz External Scaler por gRPC. El número objetivo de réplicas se calcula como:

$$r\_target = ceil((RPS\_pred \cdot f\_safety) / rps\_c)$$

donde RPS_pred es la predicción de solicitudes por segundo a un horizonte de 20 pasos de 15 segundos (5 minutos), f_safety es el factor de seguridad y rps_c representa la capacidad empírica por réplica, derivada del modelo de colas M/D/c. La estrategia híbrida aplica una regla de máximo: nunca asigna menos réplicas que las recomendadas por la guarda reactiva de CPU.

$$r\_hybrid = max(r\_predictive, r\_cpu\_guard)$$

## 4. Diseño experimental

### 4.1. Plataforma e instrumentación

Los ensayos se ejecutaron sobre un clúster dedicado de tres servidores AMD Ryzen 9 5950X, con 16 núcleos por nodo, 32 GB de memoria DDR4-3200, SMT desactivado y gobernador de frecuencia en modo performance. Esta configuración buscó reducir la variabilidad externa y mantener la frecuencia estable durante las mediciones.

El software de base incluyó k3s 1.29 como distribución de Kubernetes, Prometheus 2.47 para recolección de métricas, KEDA 2.12 para autoescalado y k6 0.50 como generador de carga. El sistema bajo prueba ejecutó inferencia de clasificación de imágenes con ResNet-50 sobre ONNX Runtime. Esta carga es representativa de servicios de inferencia en producción y, además, impone presión sobre la memoria debido a que cada instancia requiere aproximadamente 1 GiB.

### 4.2. Series experimentales

La evaluación incluyó 81 experimentos válidos, distribuidos en dos series complementarias. La Serie C.2 reunió 36 experimentos: cuatro estrategias, tres perfiles de carga y tres repeticiones. El predictor utilizado fue la versión 2.1.0, con factor de seguridad 1,2. Los perfiles incluyeron una carga gradual sinusoidal, una carga en ráfagas y un pico breve de tipo flash crowd.

La Serie C.4 reunió 45 experimentos: las cuatro estrategias anteriores más hpa70, también con tres perfiles de carga y tres repeticiones. En esta serie las trazas se escalaron a 120 RPS máximos y se limitó el sistema a 16 instancias, con el objetivo de operar dentro de la zona estable del modelo M/D/c. El predictor utilizado fue la versión 2.2.0, con factor de seguridad 1,3.

### 4.3. Protocolo estadístico

El análisis se realizó con un protocolo no paramétrico de cuatro etapas: prueba ómnibus de Kruskal-Wallis, comparaciones por pares con Mann-Whitney U, corrección de Holm-Bonferroni y tamaño de efecto mediante δ de Cliff con intervalos de confianza por remuestreo. Este enfoque evita suponer normalidad y resulta adecuado para muestras experimentales pequeñas.

Como parte de la curación de datos se eliminaron las primeras cuatro muestras de cada serie temporal, equivalentes a 60 segundos, para evitar el efecto de arranque. La decisión se validó con la regla MSER-5. Ningún experimento fue excluido del análisis final.

**Tabla 2:** Mediana [IQR] de métricas energéticas y de calidad de servicio por estrategia y serie. P95: percentil 95 de latencia; Viol. SLA: proporción de solicitudes con P95 > 500 ms o timeout.

| Estrategia | J/req | P95 (ms) | Viol. SLA (%) | Pod-seconds |
|---|---|---|---|---|
| **Serie C.2** | | | | |
| *fixed* | 1,990 [0,061] | 89,16 [2,27] | 20,82 [51,40] | 41.760 [0] |
| *hpa50* | 1,855 [0,234] | 79,67 [16,86] | 47,38 [28,24] | 30.143 [7.688] |
| *predictive* | 1,697 [0,201] | 73,80 [1,16] | 50,51 [26,45] | 21.323 [6.000] |
| *hybrid* | 1,729 [0,082] | 73,84 [0,84] | 48,59 [27,85] | 24.285 [7.725] |
| **Serie C.4** | | | | |
| *fixed* | 2,413 [0,157] | 76,38 [2,59] | 0,00 [0,00] | 27.840 [0] |
| *hpa50* | 2,198 [0,087] | 73,79 [0,05] | 40,47 [13,81] | 17.183 [3.135] |
| *hpa70* | 2,015 [0,212] | 73,75 [0,02] | 48,38 [21,77] | 12.248 [1.035] |
| *predictive* | 2,228 [0,142] | 73,75 [0,01] | 32,34 [2,29] | 20.055 [2.625] |
| *hybrid* | 2,247 [0,150] | 73,75 [0,01] | 9,95 [1,77] | 25.883 [2.655] |

## 5. Resultados

### 5.1. Ahorro energético frente al aprovisionamiento fijo

En la Serie C.2, la prueba de Kruskal-Wallis detectó diferencias significativas en J/req entre estrategias ($H = 26,8$; $p < 0,001$). Las comparaciones por pares indican que las tres estrategias dinámicas se diferencian del aprovisionamiento fijo con efecto máximo ($\delta = 1,000$; $p = 0,0025$).

**Tabla 3:** Comparaciones por pares de J/req en la Serie C.2. Mann-Whitney U con corrección Holm-Bonferroni y δ de Cliff con IC 95%.

| Grupo A | Grupo B | δ de Cliff | IC 95% | p (Holm) | Sig. |
|---|---|---|---|---|---|
| *fixed* | *hpa50* | 1,000 | [1,000; 1,000] | 0,0025 | Sí |
| *fixed* | *predictive* | 1,000 | [1,000; 1,000] | 0,0025 | Sí |
| *fixed* | *hybrid* | 1,000 | [1,000; 1,000] | 0,0025 | Sí |
| *hpa50* | *predictive* | 0,556 | [0,360; 0,751] | 0,156 | No |
| *hpa50* | *hybrid* | 0,383 | [0,168; 0,597] | 0,371 | No |
| *predictive* | *hybrid* | −0,185 | [−0,416; 0,046] | 0,537 | No |

La estrategia predictiva fue la más eficiente de la Serie C.2: 1,697 J/req frente a 1,990 J/req del aprovisionamiento fijo, lo que equivale a una reducción de 14,7%. También redujo el uso de instancias: 21.323 pod-seconds frente a 41.760, es decir, 49% menos. En el régimen evaluado, la relación observada sugiere una caída aproximada de 3% en J/req por cada 10% de reducción en pod-seconds.

El ahorro no se obtuvo a costa de mayor latencia. En la misma serie, la estrategia predictiva alcanzó P95 = 73,80 ms frente a 89,16 ms del esquema fijo, una mejora del 17,2%. Esto muestra que un escalado más ajustado puede reducir simultáneamente desperdicio energético y contención operativa.

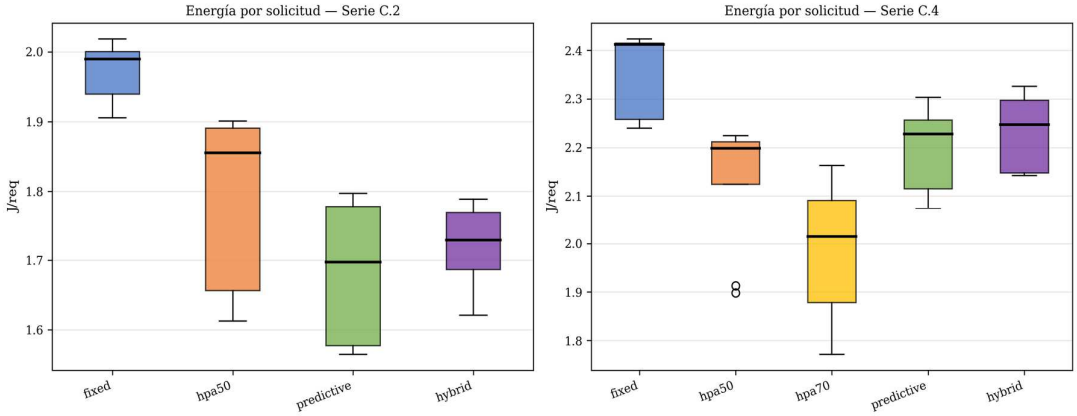


**Figura 1:** Distribución de energía por solicitud (J/req) por estrategia. Izquierda: Serie C.2. Derecha: Serie C.4, operando en zona estable del modelo M/D/c.

### 5.2. Zona estable y estrategia híbrida

La Serie C.4 modifica deliberadamente las condiciones de carga para operar en una zona estable del modelo de colas, con un máximo de 16 instancias. En este escenario, hpa70 logra el menor consumo energético (2,015 J/req), pero exhibe la peor calidad de servicio: 48,38% de violaciones. Este resultado confirma que consumir menos energía no implica, por sí solo, operar mejor.

La estrategia híbrida ofrece el balance más sólido para producción. Consume 2,247 J/req, apenas 2,2% más que hpa50, pero reduce las violaciones de nivel de servicio de 40,47% a 9,95%. La relación entre costo y beneficio es claramente asimétrica: se paga un incremento energético marginal para obtener una mejora de disponibilidad del orden del 75%.

Este comportamiento se explica por el diseño de la política híbrida. La predicción prepara recursos antes del pico, mientras que la guarda reactiva actúa cuando el predictor subestima la carga. Así, el sistema evita tanto el sobreaprovisionamiento permanente como la falta de capacidad ante ráfagas inesperadas.

### 5.3. Compromiso entre energía y disponibilidad

Los resultados distinguen dos puntos de operación. El primero es el punto de máxima eficiencia, representado por la estrategia predictiva pura: minimiza J/req y pod-seconds, por lo que es atractiva cuando los acuerdos de nivel de servicio son flexibles. El segundo es el punto orientado a producción, representado por la estrategia híbrida: sacrifica una pequeña fracción de energía para proteger la disponibilidad.

La Figura 2 resume ese compromiso. El tamaño de cada marcador representa los pod-seconds, por lo que se visualiza simultáneamente la energía, la latencia y el uso de instancias. La lectura operativa es directa: para entornos donde una violación de SLA tiene costo económico o reputacional, hybrid domina a hpa50 porque mejora fuertemente la disponibilidad con una penalización energética reducida.

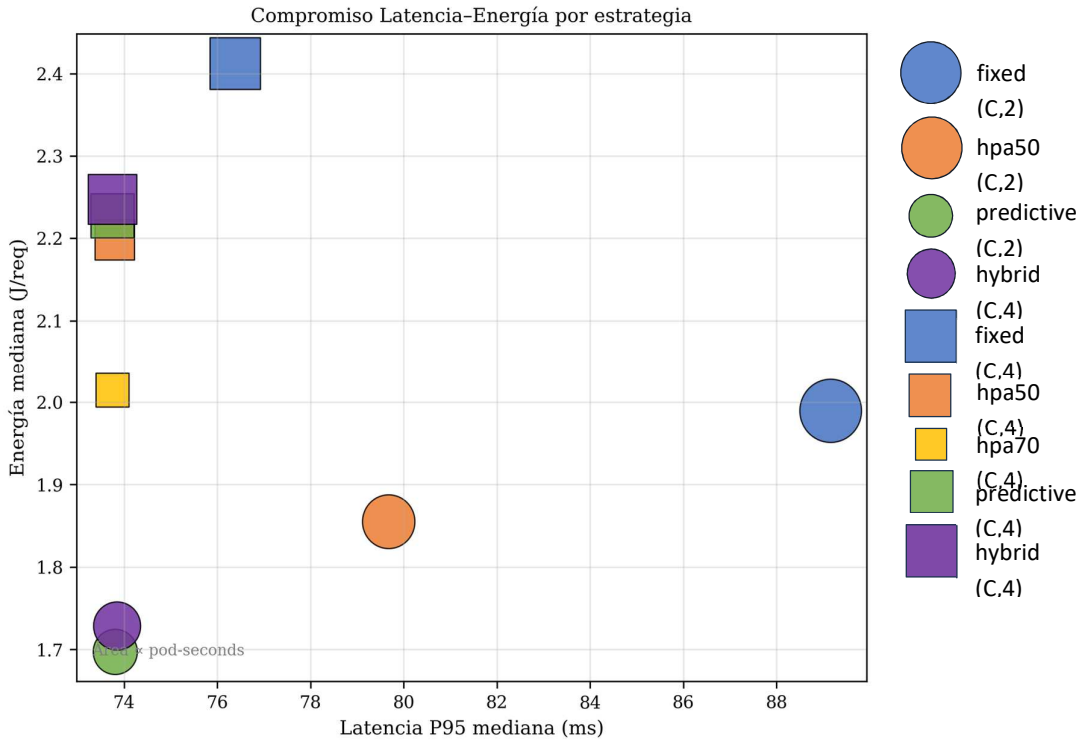


**Figura 2:** Compromiso latencia-energía por estrategia y serie. El tamaño del marcador es proporcional a los pod-seconds.

La potencia promedio muestra una tendencia coherente con el uso de instancias. Las estrategias que sostienen menos réplicas reducen la potencia total, aunque esta mejora debe interpretarse junto con las violaciones de servicio. En otras palabras, el ahorro energético solo es valioso si el sistema sigue cumpliendo su función.

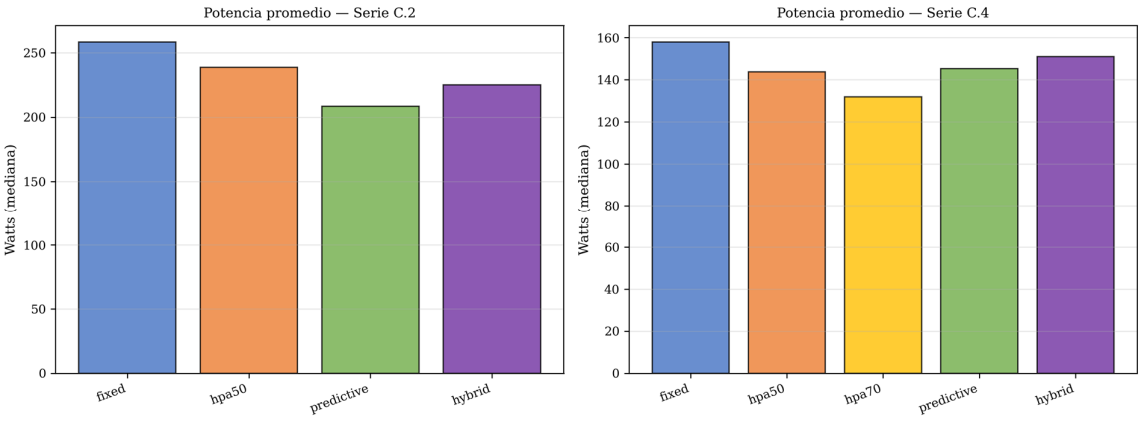


**Figura 3:** Potencia promedio por estrategia y serie. Las estrategias con menos instancias tienden a consumir menos potencia, pero pueden degradar la calidad de servicio.

### 5.4. Precisión operacional del predictor

La precisión del predictor en operación fue menor que la observada durante el entrenamiento. Aunque el modelo obtuvo MAPE = 7,38% sobre el conjunto de evaluación sintético, en los nueve experimentos predictivos de la Serie C.2 la media global del MAPE alcanzó 79,9%. Al restringir el análisis al régimen estable, la media desciende a 27,3% y la mediana a 19,4%.

Esta diferencia no invalida el enfoque, pero sí muestra una lección importante: el MAPE global puede ser engañoso cuando incluye fases de arranque y transición. En intervalos de bajo tráfico, una diferencia absoluta pequeña entre RPS real y predicho puede transformarse en un error porcentual muy alto. Por ello, para sistemas de autoescalado resulta más informativo reportar precisión por régimen operacional.

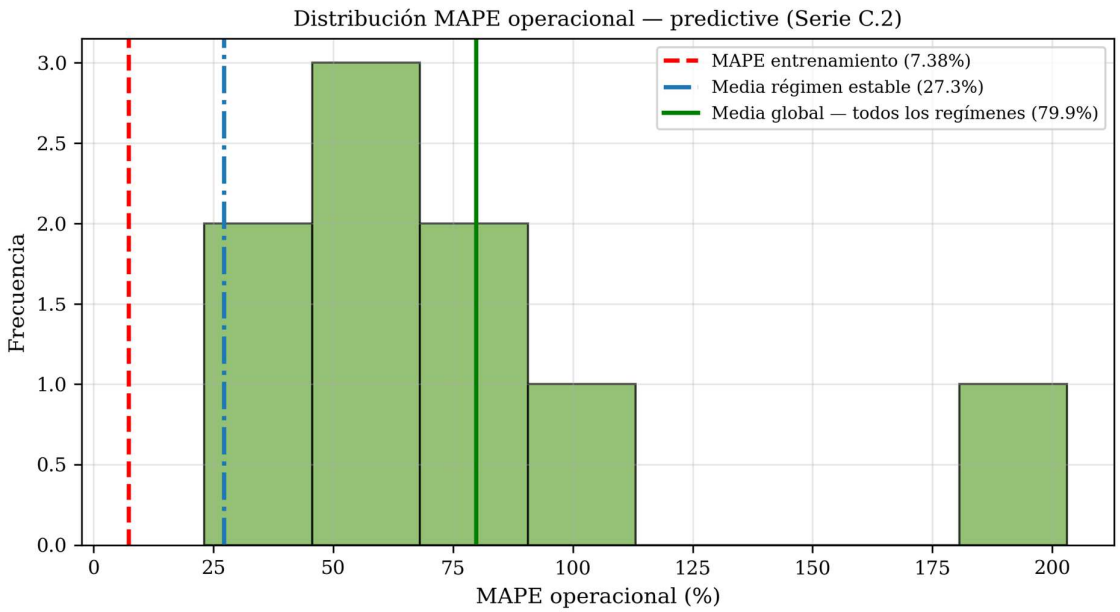


**Figura 4:** Distribución del MAPE operacional de la estrategia predictiva en la Serie C.2. La cola derecha se explica por errores porcentuales elevados durante fases de bajo volumen o transición.

En la estrategia híbrida, parte de ese error se transforma en margen de seguridad. Cuando la predicción sobreestima la demanda, el sistema sostiene algunas instancias adicionales; cuando la subestima, la guarda reactiva evita que el número de réplicas caiga por debajo del piso operativo.

## 6. Discusión

El resultado más relevante es que el autoescalado predictivo produce ahorros energéticos medibles en un clúster Kubernetes real. La reducción de 14,7% en J/req frente al aprovisionamiento fijo, acompañada por una disminución de 49% en pod-seconds, confirma que anticipar la demanda permite utilizar mejor la infraestructura.

No obstante, la estrategia más eficiente no siempre es la más conveniente. En producción, donde la disponibilidad suele tener prioridad, la alternativa híbrida ofrece una relación costo-beneficio más robusta. Su sobrecosto energético frente a hpa50 es de solo 2,2%, mientras que

la reducción de violaciones de SLA supera el 75%. Esta asimetría justifica recomendar hybrid cuando existen acuerdos de nivel de servicio estrictos.

La comparación con trabajos previos también refuerza el aporte del estudio. Mientras muchos enfoques predictivos reportan mejoras en latencia o reducción de violaciones, este trabajo mide explícitamente J/req y lo vincula con métricas de calidad de servicio. Esa integración es necesaria para evaluar sistemas de nube en un contexto donde la sostenibilidad y el costo energético tienen cada vez más peso.

Las principales limitaciones son cuatro. Primero, el hardware AMD utilizado no expone el dominio RAPL de DRAM, por lo que el consumo de memoria no se mide directamente. Segundo, se evaluó un único sistema bajo prueba, basado en inferencia de imágenes. Tercero, la granularidad de Kepler, de 15 segundos, puede suavizar picos breves de potencia. Cuarto, J/req usa como denominador todas las solicitudes emitidas, incluyendo las que terminaron en timeout o error; por ello, futuras evaluaciones deberían incorporar también J/req exitoso.

## 7. Conclusiones y trabajos futuros

La evaluación experimental confirma que el autoescalado predictivo basado en LSTM puede reducir el consumo energético por solicitud en Kubernetes. En la Serie C.2, la estrategia predictiva alcanzó una reducción de 14,7% respecto del aprovisionamiento fijo, con efecto estadístico máximo y 49% menos de uso de instancias.

El hallazgo operativo más importante es la conveniencia de la estrategia híbrida para producción. Al combinar predicción con una guarda reactiva de CPU, reduce las violaciones de nivel de servicio de 40,47% a 9,95% respecto de hpa50, con un incremento energético de apenas 2,2%. Por ello, predictive resulta adecuado cuando la prioridad principal es la eficiencia energética, mientras que hybrid es preferible cuando la disponibilidad y el cumplimiento de SLA son críticos.

Como trabajos futuros se propone repetir la evaluación en procesadores Intel con medición completa de DRAM, ampliar el estudio a cargas heterogéneas CPU-bound e I/O-bound, incorporar métricas de energía por solicitud exitosa y explorar políticas adaptativas que optimicen simultáneamente energía, latencia y disponibilidad sobre un frente de Pareto.

## Referencias

[1] Masanet, E. et al. Recalibrating global data center energy-use estimates. Science, 367(6481), 984-986, 2020.

[2] Kubernetes. Horizontal Pod Autoscaling. Documentación oficial, 2024. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[3] Dang-Quang, N. y Yoo, M. Deep learning-based autoscaling using bidirectional long short-term memory for Kubernetes. Applied Sciences, 11(9), 3835, 2021.

[4] Guruge, P. y Priyadarshana, Y. Time series forecasting-based Kubernetes autoscaling using Facebook Prophet and Long Short-Term Memory. Frontiers in Computer Science, 7, 1509165, 2025.

[5] Beloglazov, A. et al. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines. Concurrency and Computation: Practice and Experience, 24(13), 1397-1420, 2012.

[6] Mastroianni, C. et al. A probabilistic model for auto-scaling of applications in cloud environments. IEEE Transactions on Cloud Computing, 1(2), 1, 2013.

[7] Amaral, M. et al. Kepler: Kubernetes-based Efficient Power Level Exporter. Linux Foundation Energy, 2023. https://sustainable-computing.io/

[8] David, H. et al. RAPL: Memory power estimation and capping. ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 189-194, 2010.

[9] Raffin, E. et al. RAPL in practice: Limitations and workarounds. Energies, 18(2), 278, 2025.

[10] Vu, D. et al. Predictive hybrid autoscaling for containerized applications. IEEE Access, 10, 109768-109778, 2022.

[11] Santos, J. et al. Gwydion: Efficient auto-scaling for complex containerized applications in Kubernetes through Reinforcement Learning. Journal of Network and Computer Applications, 234, 104067, 2025.

[12] Anunziata, D. y Corti, E. Observabilidad de la eficiencia energética en servidores de centros de datos con solución de código abierto. CACIC 2024.

[13] Anunziata, D. y Corti, E. Evaluación comparativa del rendimiento y consumo energético en sistemas multi-core bajo distintos gobernadores de frecuencia. MECOM 2025.

[14] Anunziata, D., Corti, E. y Carossio, C. Optimización dinámica del rendimiento y eficiencia energética en sistemas multi-core bajo DVFS y contenerización. CLICAP 2026.

[15] Thompson, J. Predictive Horizontal Pod Autoscaler, 2022. https://github.com/jthomperoo/predictive-horizontal-pod-autoscaler

[16] Feng, D. y Cliff, N. The generalized Cliff's delta. Journal of Modern Applied Statistical Methods, 3(2), 287-299, 2004.

[17] Romano, J. et al. Appropriate statistics for ordinal level data. Practical Assessment, Research & Evaluation, 11(8), 1-13, 2006.