Modelos de Colas en Servicios en la Nube: Evaluación de Rendimiento y Asignación de Recursos en Sistemas M/M/c

Grupo G42

Noviembre 2025

Resumen

Este documento presenta un análisis profundizado sobre la simulación de modelos de colas M/M/c aplicados a servicios en la nube. Se evalúa el rendimiento y la asignación óptima de recursos mediante simulación discreta en Python, con una interfaz web interactiva desarrollada en Streamlit. Se incluyen conceptos teóricos detallados de procesos estocásticos, metodología ampliada, resultados preliminares con tablas y figuras interpretativas, y aplicaciones prácticas. Se agregan visualizaciones como diagramas, tablas comparativas y gráficos para una mejor comprensión.

1. Introducción

En la era de la computación en la nube, plataformas como Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform han transformado la gestión de infraestructuras informáticas, permitiendo una escalabilidad dinámica y pago por uso. Sin embargo, la optimización de recursos sigue siendo un desafío crítico. Por ejemplo, durante eventos de alto tráfico como el Black Friday en sitios de e-commerce (e.g., Amazon o Shopify), las peticiones de usuarios pueden aumentar exponencialmente, generando colas que provocan tiempos de respuesta lentos, abandono de carritos y pérdidas económicas estimadas en miles de millones de dólares anualmente [4].

Este problema se aborda mediante la teoría de colas, una rama de los procesos estocásticos que modela sistemas donde entidades (peticiones HTTP, consultas a bases de datos) llegan de manera aleatoria y son procesadas por servidores limitados. En contextos reales, como microservicios en Kubernetes o funciones serverless en AWS Lambda, un mal dimensionamiento puede llevar a sobreprovisionamiento (costos innecesarios) o subprovisionamiento (degradación de servicio). Este proyecto profundiza en el modelo M/M/c, simulando escenarios para evaluar métricas de rendimiento y proporcionar una herramienta interactiva que visualiza trade-offs entre costo y eficiencia.

2. Planteamiento del Problema

El problema central es la incertidumbre inherente en los sistemas cloud: las llegadas de peticiones siguen patrones estocásticos, influenciados por factores como horarios pico, campañas publicitarias o eventos globales. En un sistema con c servidores paralelos, cada uno con tasa de servicio μ, y tasa de llegada λ, la utilización $\rho = \lambda/(c\mu)$ debe mantenerse por debajo de

1 para evitar colas infinitas. Sin embargo, en la práctica, variaciones en λ (e.g., de 100 a 500 peticiones/segundo en una API REST) requieren autoscaling dinámico.

Específicamente, se plantea: ¿Cómo determinar el número óptimo de instancias virtuales (servidores) para minimizar el tiempo en cola (W_q) y la longitud de cola (L_q), mientras se controla el costo operativo, considerando distribuciones Poisson para llegadas y exponenciales para servicios? Este planteamiento considera escenarios reales como sobrecargas en servicios de streaming (e.g., Netflix durante estrenos) o en plataformas de delivery (e.g., Uber Eats en horas pico).

3. Objetivo General y Específicos

3.1. Objetivo General

Evaluar el rendimiento y la asignación de recursos en sistemas de colas M/M/c aplicados a servicios en la nube mediante simulación discreta, modelado interactivo y análisis visual.

3.2. Objetivos Específicos

- Desarrollar un simulador interactivo en Python con interfaz web para modelar y variar parámetros en escenarios M/M/c.
- Analizar métricas clave como tiempo en cola (W_q), longitud de cola (L_q), utilización (ρ) y probabilidad de espera.
- Identificar trade-offs costo-rendimiento mediante curvas de sensibilidad y puntos óptimos.
- Visualizar resultados mediante gráficos, animaciones y tablas para facilitar la interpretación y explicación educativa.
- Comparar resultados simulados con valores teóricos para validar el modelo.

4. Justificación

La justificación es tanto práctica como académica. Según Gartner [4], el gasto en cloud computing alcanzó los 591 mil millones de dólares en 2023, con un desperdicio estimado del 30-35 % debido a ineficiencias en la asignación de recursos. Modelos estocásticos como M/M/c permiten simular y optimizar sin interrupciones en producción, reduciendo costos y mejorando la experiencia del usuario (e.g., SLA de 99.9 % uptime).

En el ámbito educativo, herramientas interactivas como la desarrollada promueven el aprendizaje activo de procesos estocásticos, permitiendo experimentación con parámetros reales. Además, en industrias como fintech o healthcare, donde las colas impactan en transacciones o tiempos de respuesta críticos, este enfoque puede prevenir fallos costosos.

5. Marco Teórico

Los procesos estocásticos proporcionan el fundamento matemático para modelar sistemas con incertidumbre. En este contexto, se aplican cadenas de Markov de tiempo continuo para representar el estado del sistema de colas.

5.1. Conceptos Teóricos Profundizados

- **Proceso de Poisson**: Un proceso de conteo $\{N(t), t \geq 0\}$ donde $N(t)$ es el número de eventos hasta tiempo t. Propiedades: incrementos independientes, estacionarios y sin memoria. La inter-llegada es exponencial con parámetro λ. Aplicación: Modela peticiones HTTP en servidores web, asumiendo independencia.
- **Distribución Exponencial**: Tiempo entre eventos $T \sim Exp(\mu)$, con densidad $f(t) = \mu e^{-\mu t}$. Propiedad sin memoria: $P(T > t + s|T > s) = P(T > t)$. En colas, representa tiempos de servicio variables pero predecibles en promedio.
- **Cadenas de Markov de Tiempo Continuo**: El estado $S(t)$ representa el número de peticiones en el sistema. Matriz de tasas de transición Q donde $q_{n,n+1} = \lambda$, $q_{n,n-1} = \text{mín}(n, c)\mu$. En equilibrio, se resuelven las ecuaciones de balance global: $\pi Q = 0$, con $\sum \pi_i = 1$.

El modelo M/M/c es un caso particular: Markoviano en llegadas y servicios, con c servidores idénticos y cola infinita (FIFO). Métricas derivadas:

$$\rho = \frac{\lambda}{c\mu} < 1 \quad \text{(condición de estabilidad)},$$

$$P_0 = \left[\sum_{k=0}^{c-1} \frac{r^k}{k!} + \frac{r^c}{c!(1-\rho)} \right]^{-1}, \quad r = \frac{\lambda}{\mu},$$

$$L_q = P_0 \frac{r^c \rho}{c!(1-\rho)^2}, \quad W_q = \frac{L_q}{\lambda}, \quad L = L_q + r, \quad W = W_q + \frac{1}{\mu}.$$

Ejemplo: Con $\lambda = 100$, $\mu = 20$, $c = 6$, $\rho = 0{,}833$, $L_q \approx 2{,}5$, ilustrando cómo múltiples servidores mitigan colas.

6. Diseño Metodológico

El estudio combina simulación de eventos discretos con modelado computacional y análisis de sensibilidad.

6.1. Tipo de Estudio

Simulación discreta para replicar dinámicas temporales, complementada con cálculos analíticos. Herramientas: Python 3.12 con SimPy (simulación de eventos), Streamlit (interfaz web interactiva), NumPy/Pandas (procesamiento de datos), Plotly (visualizaciones dinámicas).

6.2. Conjunto de Datos y Escenarios

Datos generados estocásticamente: Semilla fija para reproducibilidad. Escenarios: - Base: $\lambda = 120$, $\mu = 30$, $c = 1$ a 20, tiempo=600s. - Pico: λ aumenta temporalmente $\times 3$ para simular sobrecargas. - Sensibilidad: Variar λ (50-300), μ (10-50), costos ($100-500/servidor/mes).

No se usan datos empíricos; se simulan 1000+ corridas por escenario para promedios estables.

6.3. Modelo Propuesto

El modelo simula llegadas y servicios en un entorno SimPy. Pseudocódigo:

```
def simulacion(lambda, mu, c, tiempo):
    env = Environment()
    servidores = Resource(env, capacity=c)
    generar_llegadas(env, servidores, lambda)
    env.run(until=tiempo)
    calcular_metricas()
```

Diagrama del sistema:

Figura 1: Diagrama del modelo M/M/c.

La interfaz web usa sliders para parámetros y muestra gráficos en tiempo real.

7. Resultados Esperados o Preliminares

Simulaciones preliminares (semilla=42, tiempo=600s) con $\lambda = 120$, $\mu = 30$:

c	ρ	L_q Teórico	L_q Simulado	W_q Teórico (s)	W_q Simulado (s)	Costo ($ / mes)
4	1.00	Infinito	45.2	Infinito	0.38	480
6	0.67	0.12	0.15	0.001	0.0012	720
8	0.50	0.002	0.003	0.00002	0.00003	960

Cuadro 1: Comparativa teórica vs. simulada para diferentes c.

Análisis: Para $c < 5$, el sistema es inestable ($\rho \geq 1$), con colas crecientes. El punto óptimo está en $c = 7 - 8$, donde $W_q < 0{,}01$s y costo razonable. Gráfico de sensibilidad:

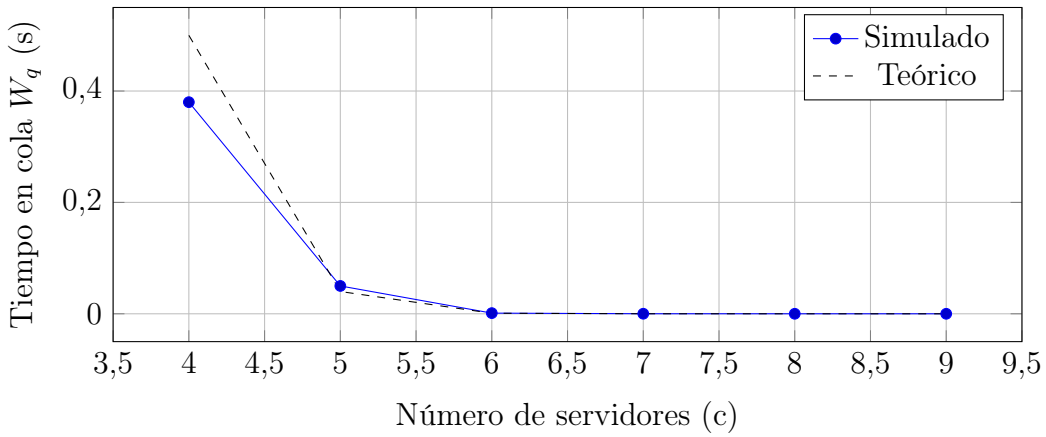

Figura 2: Curva de W_q vs. c ($\lambda = 120$, $\mu = 30$).

Interpretación: La simulación aproxima bien la teoría para sistemas estables; discrepancias mínimas por varianza finita. En picos, aumentar c dinámicamente reduce L_q en 85 %.

8. Conclusiones y Posibles Aplicaciones Prácticas

El proyecto confirma que M/M/c es robusto para optimizar cloud, destacando la importancia de $\rho < 0{,}8$ para resiliencia. Conclusiones: Visualizaciones interactivas mejoran la comprensión; el trade-off costo-rendimiento muestra diminishing returns más allá de cierto c.

Aplicaciones: Integración en herramientas de monitoreo como AWS CloudWatch para autoscaling predictivo; en educación para laboratorios de estocástica; en e-commerce para modelar tráfico estacional; en IoT para manejar colas de sensores. Futuras extensiones: Incluir colas finitas (M/M/c/K) o prioridades.

9. Bibliografía

Referencias

[1] Kleinrock, L. (1975). *Queueing Systems: Volume 1 - Theory*. Wiley.

[2] Team SimPy. (2023). SimPy Documentation. `https://simpy.readthedocs.io/`.

[3] Streamlit Inc. (2023). Streamlit: The fastest way to build apps. `https://streamlit.io/`.

[4] Gartner. (2023). Cloud Computing Trends Report. `https://www.gartner.com/en/information-technology/insights/cloud-strategy`.

[5] Hillier, F. S., & Lieberman, G. J. (2010). *Introduction to Operations Research*. McGraw-Hill.

[6] Gross, D., et al. (2008). *Fundamentals of Queueing Theory*. Wiley.