INSTITUTO FEDERAL
Santa Catarina

Exercício - Filas

Exercício Avaliativo M/M/1 e M/M/c

Curso: Engenharia de Telecomunicações
Disciplina: ADS029009 - Avaliação de desempenho de sistemas
Professor: Eraldo Silveira e Silva

Alunos
Leonardo Ludvig Silva
Jessica Gomes Carico

Junho de 2025

1 Introdução

Este relatório apresenta a análise de um exercício proposto pelo professor da disciplina. O objetivo é avaliar diferentes configurações de servidores para um sistema de processamento de imagens científicas.

1.1 Cenário proposto

Durante um evento especial, esse sistema passou a receber uma carga média de **150 imagens por minuto**, enquanto o servidor atual possui capacidade de processar **180 imagens por minuto**. Com isso, foi solicitado analisar, em termos de **tempo médio no sistema** e **tempo médio na fila**, se seria melhor:

- Manter a configuração atual (um servidor com capacidade de 180 img/min);
- Distribuir a carga entre 5 servidores independentes (cada um com 40 img/min e carga de 30 img/min);
- Utilizar uma fila única com 5 processadores (cada um com capacidade de 40 img/min).

O exercício também pediu o cálculo da **utilização do sistema** e a apresentação dos resultados em forma de **tabela comparativa**, seguida de uma conclusão sobre a melhor alternativa.

2 Desenvolvimento do código - sem IA

Para analisar o desempenho do sistema em cada configuração, utilizamos o modelo de filas M/M/1 e M/M/c da Teoria das Filas. As principais fórmulas utilizadas foram:

- Número médio de clientes no sistema:

$$E_N = \frac{\lambda}{\mu - \lambda}$$

- Tempo médio no sistema (utilizando o conceito de Lei de Little):

$$E_R = \frac{E_N}{\lambda}$$

- Tempo médio de espera na fila:

$$E_{Wq} = \frac{\lambda}{\mu(\mu - \lambda)}$$

- Utilização do sistema:

$$\rho = \frac{\lambda}{\mu}$$

Código em Python

O código a seguir foi utilizado para aplicar essas fórmulas e obter os tempos médios de resposta e espera para os três cenários:

```
# Cenário atual
mi = 180
lambdac = 150

e_n = lambdac / (mi - lambdac)
e_r = e_n / lambdac
e_wq = lambdac / (mi * (mi - lambdac))
ro = lambdac / mi

# Alternativa (i): 5 servidores independentes
lambda1 = 30
mi1 = 40
e_n1 = lambda1 / (mi1 - lambda1)
e_r1 = e_n1 / lambda1
e_wq1 = lambda1 / (mi1 * (mi1 - lambda1))
ro1 = lambda1 / mi1

# Alternativa (ii): Fila única com 5 servidores
c = 5
lambda2 = 30 * c
mi2 = 40 * c
e_n2 = lambda2 / (mi2 - lambda2)
e_r2 = e_n2 / lambda2
e_wq2 = lambda2 / (mi2 * (mi2 - lambda2))
ro2 = lambda2 / mi2
```

3 Comparação entre os resultados

A partir dos valores calculados para cada configuração, é possível observar algumas diferenças nos indicadores de desempenho do sistema. O cenário atual, com apenas um servidor, apresenta o maior número médio de clientes no sistema e uma alta taxa de utilização. A alternativa com cinco servidores independentes reduz o número médio de clientes, mas resulta em um tempo médio no sistema mais elevado, devido ao isolamento das filas. Por outro lado, a fila única com cinco servidores apresenta o melhor desempenho geral, com menores tempos médios de espera e permanência no sistema, mantendo a utilização em um nível eficiente.

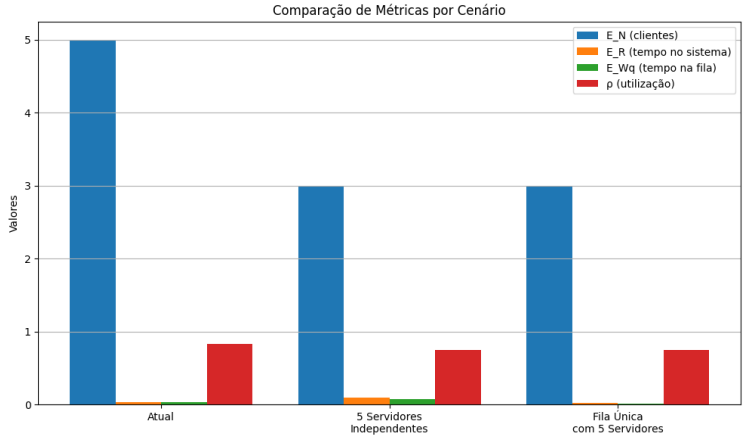
Essas diferenças podem ser visualizadas comparativamente na Figura 1, que ilustra graficamente os principais indicadores de desempenho, e também na Tabela 1, que resume os valores obtidos em cada cenário.

Tabela 1: Comparação de métricas dos cenários

Cenário	$\mathbf{E_N}$ (clientes)	$\mathbf{E_R}$ (tempo no sistema)	$\mathbf{E_{Wq}}$ (tempo na fila)	ρ (utilização)
Atual	5.00	0.0333	0.0278	0.8333
5 Servidores Independentes	3.00	0.1000	0.0750	0.7500
Fila Única com 5 Servidores	3.00	0.0200	0.0150	0.7500

Fonte: Elaborado pelos autores

Figura 1: Gráfico de barras da comparação de métricas por cenário



Fonte: Elaborado pelos autores

4 Código gerado por IA

```python
import math
import pandas as pd

def mm1_metrics(lambd, mu):
    rho = lambd / mu
    W = 1 / (mu - lambd)
    Wq = lambd / (mu * (mu - lambd))
    Ws = 1 / mu
    return rho, W, Wq, Ws

def mmc_metrics(lambd, mu, c):
    rho = lambd / (c * mu)
    sum_terms = sum((lambd / mu) ** n / math.factorial(n) for n in range(c))
    last_term = ((lambd / mu) ** c) / (math.factorial(c) * (1 - rho))
    P0 = 1 / (sum_terms + last_term)
    Pw = last_term * P0
    Wq = Pw * (1 / mu) / (c * (1 - rho))
    W = Wq + 1 / mu
    Ws = 1 / mu
    return rho, W, Wq, Ws

lambda_total = 150
mu_atual = 180

rho_1, W_1, Wq_1, Ws_1 = mm1_metrics(lambda_total, mu_atual)
lambda_indiv = 30
mu_indiv = 40
rho_i, W_i, Wq_i, Ws_i = mm1_metrics(lambda_indiv, mu_indiv)
c = 5
mu_multi = 40
rho_ii, W_ii, Wq_ii, Ws_ii = mmc_metrics(lambda_total, mu_multi, c)

df = pd.DataFrame({
    "Configuração": ["Atual (1 servidor)", "5 Servidores M/M/1", "Fila Única M/M/5"],
    "Utilização ()": [rho_1, rho_i, rho_ii],
    "Tempo total no sistema (W) [min]": [W_1, W_i, W_ii],
    "Tempo na fila (Wq) [min]": [Wq_1, Wq_i, Wq_ii],
    "Tempo de atendimento (Ws) [min]": [Ws_1, Ws_i, Ws_ii]
})

print(df.round(4))
```

Tabela 2: Comparação entre as configurações de servidores

Configuração	**Utilização (ρ)**	**Tempo total no sistema (W) [min]**	**Tempo na fila (Wq) [min]**
Atual (1 servidor)	0.8333	0.0333	0.0278
5 Servidores M/M/1	0.7500	0.1000	0.0750
Fila Única M/M/5	0.7500	0.0342	0.0092

Observa-se que os valores calculados para as três configurações estão apresentados na Tabela 2. Nota-se que, no caso da fila única com múltiplos servidores (M/M/5), os valores obtidos foram diferente mas coerentes em relação a proporção, isso porquê utilizamos a mesma fórmula de m/m/1 para o cáculo com multiplos servidores, mas na realidade é preciso aplicar as seguintes expressões:

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

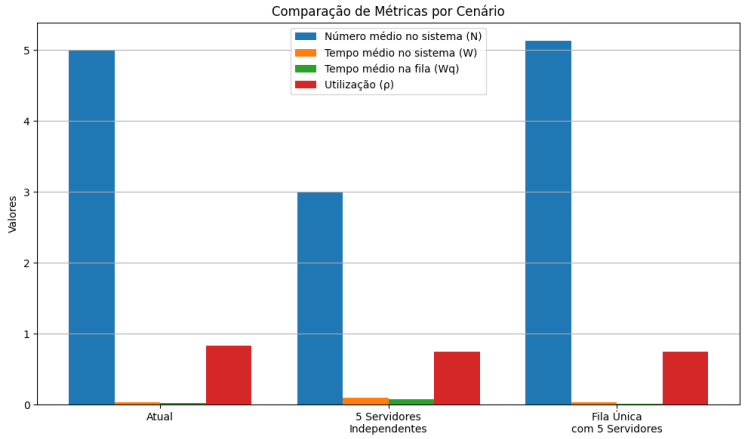
$$P_{fila} = \frac{\left(\frac{\lambda}{\mu}\right)^c}{c!\,(1-\rho)} \cdot P_0 \tag{2}$$

$$E[W_q] = \frac{P_{fila}}{c\mu - \lambda} \tag{3}$$

$$E[W] = E[W_q] + \frac{1}{\mu} \tag{4}$$

Corrigindo esse cálculo, chega-se a valores da Tabela 2 e a Figura 2.

Figura 2: Gráfico de barras da comparação de métricas por cenário



Fonte: Elaborado pelos autores

5 Conclusão

A comparação entre as configurações mostrou que o cenário atual, com um servidor único, apresenta alta utilização. A alternativa com cinco servidores independentes diminui a carga individual, mas gera filas isoladas, aumentando o tempo no sistema. Já a fila única com cinco servidores apresenta o melhor desempenho, com menor tempo de espera e utilização equilibrada apesar de ter o maior número médio de clientes no sistema. Portanto, a configuração de fila única com múltiplos servidores é a recomendada para otimizar o processamento de imagens, garantindo maior eficiência e menor tempo de resposta.

6 Referência dos Códigos

Código sem IA:

```
import matplotlib.pyplot as plt

# ********************************
# Cenário atual
mi = 180
lambdac = 150

e_n = lambdac / ( mi - lambdac)
e_r = e_n / lambdac
e_wq = lambdac / (mi * (mi - lambdac))
ro = lambdac / mi

print('E_N= ', e_n)
print('E_R (tempo médio no sistema)= ', e_r)
print('E_Wq (tempo médio na fila) = ', e_wq)
print('Utilização do sistema = ', ro)

# ********************************
# Alternativa (i): 5 servidores independentes
lambda1 = 30
mi1 = 40

e_n1 = lambda1 / (mi1 - lambda1)
e_r1 = e_n1 / lambda1
e_wq1 = lambda1 / (mi1 * (mi1 - lambda1))
ro1 = lambda1 / mi1

print('\nE_N1= ', e_n1)
print('E_R1 (tempo médio no sistema)= ', e_r1)
print('E_Wq1 (tempo médio na fila)= ', e_wq1)
print('Utilização do sistema = ', ro1)

# ********************************
# Alternativa (ii): Fila única com 5 servidores
c = 5
lambda2 = 30 * c
mi2 = 40 * c

e_n2 = lambda2 / (mi2 - lambda2)
e_r2 = e_n2 / lambda2
e_wq2 = lambda2 / (mi2 * (mi2 - lambda2))
ro2 = lambda2 / mi2

print('\nE_N2= ', e_n2)
print('E_R2 (tempo médio no sistema)= ', e_r2)
print('E_Wq2 (tempo médio na fila)= ', e_wq2)
print('Utilização do sistema = ', ro2)

# ********************************
# Gráfico comparativo
cenarios = ['Atual', '5 Servidores\nIndependentes', 'Fila Única\ncom 5 Servidores']
E_N = [e_n, e_n1, e_n2]
E_R = [e_r, e_r1, e_r2]
E_Wq = [e_wq, e_wq1, e_wq2]
rho = [ro, ro1, ro2]

bar_width = 0.2
x = range(len(cenarios))

plt.figure(figsize=(10, 6))

```

```
plt.bar([i - 1.5 * bar_width for i in x], E_N, width=bar_width, label='E_N (clientes)')
plt.bar([i - 0.5 * bar_width for i in x], E_R, width=bar_width, label='E_R (tempo no sistema)')
plt.bar([i + 0.5 * bar_width for i in x], E_Wq, width=bar_width, label='E_Wq (tempo na fila)')
plt.bar([i + 1.5 * bar_width for i in x], rho, width=bar_width, label=' (utilização)')

plt.xticks(x, cenarios)
plt.ylabel('Valores')
plt.title('Comparação de Métricas por Cenário')
plt.legend()
plt.grid(True, axis='y')
plt.tight_layout()
plt.show()
```

Código gerado pela IA:

```
import math
import pandas as pd

# ---------- Funções para M/M/1 ----------
def mm1_metrics(lambd, mu):
    rho = lambd / mu
    W = 1 / (mu - lambd)             # Tempo médio no sistema
    Wq = lambd / (mu * (mu - lambd)) # Tempo médio na fila
    Ws = 1 / mu                      # Tempo de atendimento
    return rho, W, Wq, Ws

# ---------- Função para M/M/c ----------
def mmc_metrics(lambd, mu, c):
    rho = lambd / (c * mu)

    # Cálculo da probabilidade de 0 na fila (P0)
    sum_terms = sum((lambd / mu) ** n / math.factorial(n) for n in range(c))
    last_term = ((lambd / mu) ** c) / (math.factorial(c) * (1 - rho))
    P0 = 1 / (sum_terms + last_term)

    # Probabilidade de haver espera na fila (Erlang-C)
    Pw = last_term * P0

    # Tempo médio na fila
    Wq = Pw * (1 / mu) / (c * (1 - rho))
    W = Wq + 1 / mu
    Ws = 1 / mu
    return rho, W, Wq, Ws

# ---------- Parâmetros ----------
lambda_total = 150
mu_atual = 180

# Configuração atual (1 servidor M/M/1)
rho_1, W_1, Wq_1, Ws_1 = mm1_metrics(lambda_total, mu_atual)

# Alternativa (i): 5 servidores M/M/1
lambda_indiv = 30
mu_indiv = 40
rho_i, W_i, Wq_i, Ws_i = mm1_metrics(lambda_indiv, mu_indiv)

# Alternativa (ii): Fila única M/M/5
c = 5
mu_multi = 40
rho_ii, W_ii, Wq_ii, Ws_ii = mmc_metrics(lambda_total, mu_multi, c)

# ---------- Construção da Tabela ----------
df = pd.DataFrame({
    "Configuração": ["Atual (1 servidor)", "5 Servidores M/M/1", "Fila Única M/M/5"],
    "Utilização ()": [rho_1, rho_i, rho_ii],
    "Tempo total no sistema (W) [min]": [W_1, W_i, W_ii],
```

```
    "Tempo na fila (Wq) [min]": [Wq_1, Wq_i, Wq_ii],
    "Tempo de atendimento (Ws) [min]": [Ws_1, Ws_i, Ws_ii]
})

# ---------- Resultados ----------
print(df.round(4))
```

Nosso código corrigido:

```
import matplotlib.pyplot as plt
import math

# ********************************
# Cenário atual (M/M/1)
mi = 180
lambdac = 150

e_n = lambdac / (mi - lambdac)
e_r = e_n / lambdac
e_wq = lambdac / (mi * (mi - lambdac))
ro = lambdac / mi

print('E_N= ', e_n)
print('E_R (tempo médio no sistema)= ', e_r)
print('E_Wq (tempo médio na fila) = ', e_wq)
print('Utilização do sistema = ', ro)

# ********************************
# Alternativa (i): 5 servidores independentes (cada um M/M/1)
lambda1 = 30
mi1 = 40

e_n1 = lambda1 / (mi1 - lambda1)
e_r1 = e_n1 / lambda1
e_wq1 = lambda1 / (mi1 * (mi1 - lambda1))
ro1 = lambda1 / mi1

print('\nE_N1= ', e_n1)
print('E_R1 (tempo médio no sistema)= ', e_r1)
print('E_Wq1 (tempo médio na fila)= ', e_wq1)
print('Utilização do sistema = ', ro1)

# ********************************
# Alternativa (ii): Fila única com 5 servidores (M/M/c)

c = 5
lambda2 = 150  # carga total recebida
mu = 40        # taxa de atendimento por servidor
ro2 = lambda2 / (c * mu)  # utilização do sistema

# Cálculo de P0 (probabilidade do sistema vazio)
sum_terms = sum((lambda2/mu)**n / math.factorial(n) for n in range(c))
last_term = ((lambda2/mu)**c) / (math.factorial(c) * (1 - ro2))
P0 = 1 / (sum_terms + last_term)

# Probabilidade da fila não estar vazia (Pfila)
Pfila = (((lambda2/mu)**c) / (math.factorial(c) * (1 - ro2))) * P0

# Tempo médio na fila Wq
Wq2 = Pfila / (c * mu - lambda2)

# Tempo médio no sistema W
W2 = Wq2 + 1/mu

# Número médio de clientes no sistema (Little's Law)
N2 = lambda2 * W2
```

```
58
59 print('\nAlternativa (ii) - Fila única M/M/c')
60 print(f'Utilização do sistema () = {ro2:.4f}')
61 print(f'P0 = {P0:.4f}')
62 print(f'Probabilidade fila (Pfila) = {Pfila:.4f}')
63 print(f'Tempo médio na fila (Wq) = {Wq2:.4f} min')
64 print(f'Tempo médio no sistema (W) = {W2:.4f} min')
65 print(f'Número médio de clientes no sistema (N) = {N2:.4f}')
66
67 # ********************************
68 # Gráfico comparativo
69 cenarios = ['Atual', '5 Servidores\nIndependentes', 'Fila Única\ncom 5 Servidores']
70 E_N = [e_n, e_n1, N2]
71 E_R = [e_r, e_r1, W2]
72 E_Wq = [e_wq, e_wq1, Wq2]
73 rho = [ro, ro1, ro2]
74
75 bar_width = 0.2
76 x = range(len(cenarios))
77
78 plt.figure(figsize=(10, 6))
79
80 plt.bar([i - 1.5 * bar_width for i in x], E_N, width=bar_width, label='Número médio no sistema
       (N)')
81 plt.bar([i - 0.5 * bar_width for i in x], E_R, width=bar_width, label='Tempo médio no sistema (
       W)')
82 plt.bar([i + 0.5 * bar_width for i in x], E_Wq, width=bar_width, label='Tempo médio na fila (Wq
       )')
83 plt.bar([i + 1.5 * bar_width for i in x], rho, width=bar_width, label='Utilização ()')
84
85 plt.xticks(x, cenarios)
86 plt.ylabel('Valores')
87 plt.title('Comparação de Métricas por Cenário')
88 plt.legend()
89 plt.grid(True, axis='y')
90 plt.tight_layout()
91 plt.show()
```

Referências

Referências

S. E. Silva, *Avaliação de desempenho de sistemas*, Material de aula, Disciplina ADS029009 – Engenharia de Telecomunicações, IFSC, Junho de 2025.