Implementando um Simulador de Processos de Negócio

Instituto Federal do Espírito Santo (IFES)
Bacharelado em Sistemas de Informação
Disciplina: Gerência de Processos de Negócio
Prof. Mateus Barcellos Costa

20 de janeiro de 2025

- Este simulador utiliza o **modelo BPMN** para definir processos de negócio.
- Objetivos principais:
  - Analisar o comportamento de processos de negócio.
  - Simular chegadas exponenciais negativas e tempos de execução exponenciais negativos.
  - Identificar gargalos e tempos de espera.
- Tecnologias utilizadas:
  - Biblioteca `pm4py`.
  - Python.

Antes de apresentarmos o modelo de simulação iremos ver conceitos necessários de Teoria das Filas

Introdução à Teoria das Filas

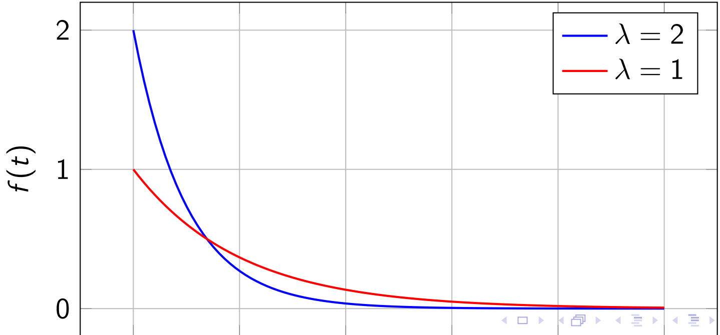
- A Teoria das Filas é usada para modelar sistemas que envolvem espera, como:
  - Call centers.
  - Sistemas de atendimento em bancos.
  - Redes de computadores.
- O modelo **M/M/1** é um dos mais simples:
  - **M/M/1**: Uma fila, chegadas e saídas exponenciais.
  - Um único servidor.

Distribuição Exponencial Negativa

- Modela o intervalo de tempo entre chegadas ou saídas.
- **Fórmula da Função Densidade de Probabilidade (PDF):**

$$f(t;\lambda) = \lambda e^{-\lambda t}, \quad t \geq 0$$

- **Características:**
  - A média é $\frac{1}{\lambda}$.
  - É uma distribuição contínua e decrescente.



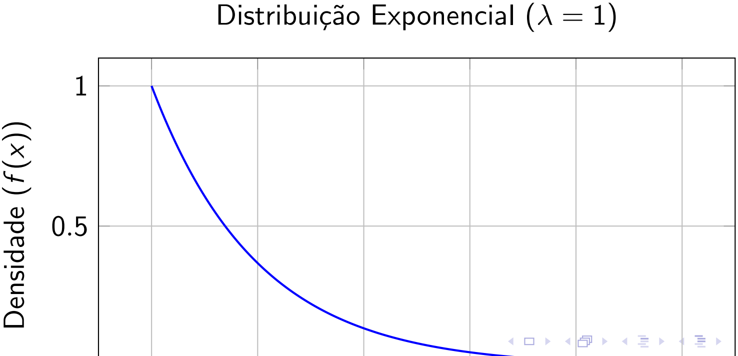
Distribuição Exponencial Negativa

Características:

- Modelo contínuo que descreve o tempo entre dois eventos consecutivos.
- A função densidade de probabilidade (PDF) é dada por:

$$f(x;\lambda) = \lambda e^{-\lambda x}, \quad x \geq 0$$

onde λ é a taxa média de eventos por unidade de tempo.

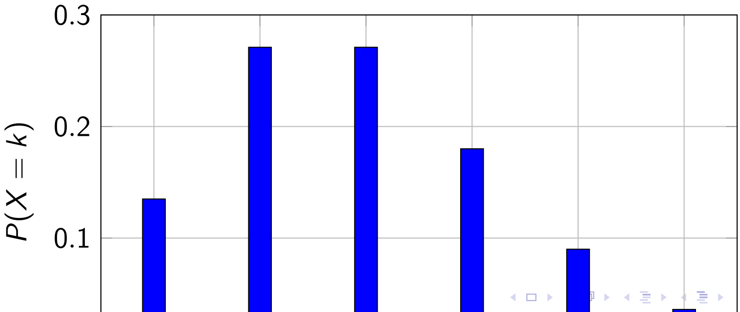


Distribuição de Poisson

- Modela o número de chegadas em um intervalo de tempo.
- **Fórmula da Probabilidade:**

$$P(X = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, \quad k = 0, 1, 2, \ldots$$

- **Características:**
  - A média e variância são λt.
  - É uma distribuição discreta.



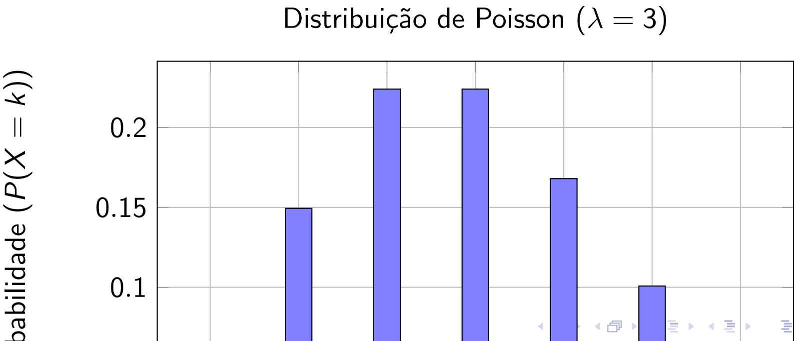
Distribuição de Poisson

Características:

- Modelo discreto que descreve o número de eventos ocorrendo em um intervalo fixo de tempo ou espaço.
- A probabilidade de ocorrer exatamente k eventos é dada por:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, 2, \ldots$$

onde λ é a taxa média de eventos.



Comparação Intuitiva

- **Distribuição de Poisson:**
  - Discreta: modela o número de eventos.
  - Exemplo: número de chegadas em um sistema em um intervalo fixo.
  - A distribuição de Poisson está no domínio da frequência, pois ela modela o número de eventos que ocorrem em um intervalo fixo de tempo ou espaço. É usada para contar eventos em um dado intervalo, ao invés de medir o tempo entre eles.
- **Distribuição Exponencial Negativa:**
  - Contínua: modela o tempo entre eventos consecutivos.
  - Exemplo: intervalo de tempo entre duas chegadas consecutivas.
  - A distribuição Exponencial Negativa está no domínio do tempo, pois ela modela o tempo entre eventos consecutivos. Em outras palavras, ela descreve quanto tempo é necessário para que um próximo evento ocorra em um processo de Poisson.

Modelo M/M/1

- **Características:**
  - Chegadas: Processo de Poisson com taxa λ.
  - Saídas: Tempo de serviço exponencial com taxa μ.
  - Fila com capacidade infinita e apenas um servidor.
- **Taxa de Utilização:**

$$\rho = \frac{\lambda}{\mu}, \quad 0 \leq \rho < 1$$

- **Probabilidade de n clientes na fila:**

$$P_n = (1 - \rho)\rho^n$$

Teorema de Little

- Relaciona medidas de desempenho de qualquer sistema estável:

$$L = \lambda W$$

  - L: Número médio de clientes no sistema.
  - λ: Taxa média de chegada.
  - W: Tempo médio de espera no sistema.
- **Exemplo:**
  - $\lambda = 10$ chegadas/minuto.
  - $W = 2$ minutos.
  - $L = \lambda W = 10 \times 2 = 20$.

Exemplo Prático

- **Cenário:**
  - Taxa de chegada $\lambda = 5$ clientes/minuto.
  - Taxa de saída $\mu = 8$ clientes/minuto.
- **Cálculos:**
  - $\rho = \frac{5}{8} = 0.625$.
  - Tempo médio no sistema: $W = \frac{1}{\mu - \lambda} = \frac{1}{8-5} = 0.33$ minutos.
  - Número médio de clientes no sistema: $L = \lambda W = 5 \times 0.33 = 1.65$.

Conclusão

- O modelo M/M/1 é uma base simples e poderosa para analisar sistemas com filas.
- O entendimento de distribuições exponencial e de Poisson é fundamental.
- O Teorema de Little fornece uma ferramenta prática para cálculos rápidos.
- Aplicações incluem desde call centers até redes de computadores.

Modelo de Simulação

- **Taxa de chegada:** Eventos chegam ao processo com tempos determinados por uma distribuição exponencial negativa.
- **Tarefas:** Cada tarefa é executada com um tempo baseado em uma distribuição de Poisson.
- **Ocupação do recurso:** Apenas um recurso por tarefa. Se ocupado, eventos aguardam na fila.
- **Medições:**
  - Tempo de espera.
  - Tempo total de execução das tarefas.
  - Tempo de conclusão de cada evento.

Estrutura do Simulador

Classe BPMNProcess: Gerenciamento do Modelo BPMN

```
class BPMNProcess:
    def __init__(self, bpmn_file):
        self.bpmn_file = bpmn_file
        self.tasks = {}
        self.load_bpmn_file()

    def load_bpmn_file(self):
        tree = ET.parse(self.bpmn_file)
        root = tree.getroot()
        ns = {'bpmn': 'http://www.omg.org/spec/BPMN
            /20100524/MODEL'}

        for task in root.findall('.//bpmn:task', ns):
            task_id = task.attrib['id']
            task_name = task.attrib.get('name', f"
                Unnamed Task ({task_id})")
            self.tasks[task_id] = {
                'name': task_name,
                'mean_time': random.randint(1, 100)

            }
```

Estrutura do Simulador

Classe ProcessSimulator: Simulação de Processos

```
class ProcessSimulator:
    def __init__(self, bpmn_process, arrival_rate,
        simulation_time):
        self.process = bpmn_process
        self.arrival_rate = arrival_rate
        self.simulation_time = simulation_time
        self.token_arrivals = []
        self.events = []

    def generate_arrivals(self):
        time = 0
        while time < self.simulation_time:
            interarrival_time = np.random.exponential(
                self.arrival_rate)
            time += interarrival_time
            if time < self.simulation_time:
                self.token_arrivals.append(time)
```

Estimativa de Tempo de Espera

- O tempo de espera é calculado com base na ocupação dos recursos.
- Para cada chegada, verifica-se:
  - O próximo recurso disponível (menor tempo de término entre ocupações).
  - A diferença entre o tempo de chegada e o próximo recurso disponível.
- **Cálculo:**

```
resource_occupancy = [end_time for end_time in resource_occupancy
                             if end_time > arrival_time]
waiting_time = max(0, min(resource_occupancy, default=arrival_time) - arrival_time)
current_time = arrival_time + waiting_time
```

Execução das Tarefas

Lógica para simular o tempo das tarefas:

```
task_durations = []
for trace in sim_result:
    for event in trace:
        task_name = event.get('concept:name', None)
        if task_name:
            task = next(
                (task_id, t) for task_id, t in self.process.tasks.items()
                if t['name'] == task_name
            )
            task_id = task[0]
            task_data = task[1]

            if task_data and task_data['mean_time'] is not None:
                duration = np.random.poisson(task_data['mean_time'])
                formatted_duration = format_time(duration)
                task_durations.append(f"{task_id}: {formatted_duration}")
```

Relatório da Simulação

- Detalhes registrados para cada chegada:
  - Tempo de chegada.
  - Tempo de conclusão.
  - Tempo total de execução das tarefas.
  - Tempo de espera.
  - Duração de cada tarefa.

Relatório da Simulação

```python
def report(self):
    total_times = [event['completion_time'] - event['arrival_time'] for event in self.events]
    print("### Simulation Report ###")
    print(f"Total Computers Processed: {len(self.events)}")
    print(f"Average Completion Time: {format_time(np.mean(total_times))}")
    for idx, event in enumerate(self.events):
        arrival_hms = format_time(event['arrival_time'])
        completion_hms = format_time(event['completion_time'])
        print(f"Computer {idx + 1}: Arrival = {arrival_hms}, Completion = {completion_hms}")
```

Descrição dos Resultados

- Os resultados sao de um modelo de processo de um SRC (reparos de computador):
  - **Arrival (Chegada):** Tempo em que o computador entra no sistema.
  - **Completion (Conclusão):** Tempo em que o processamento do computador é concluído.
  - **Total Task Time (Tempo Total de Tarefas):** Tempo gasto executando as tarefas.
  - **Waiting Time (Tempo de Espera):** Tempo que o computador ficou aguardando na fila antes de começar o processamento.
- Objetivo: Visualizar graficamente o comportamento do sistema e os tempos de espera.

Gráfico 1: Tempo de Conclusão por Computador

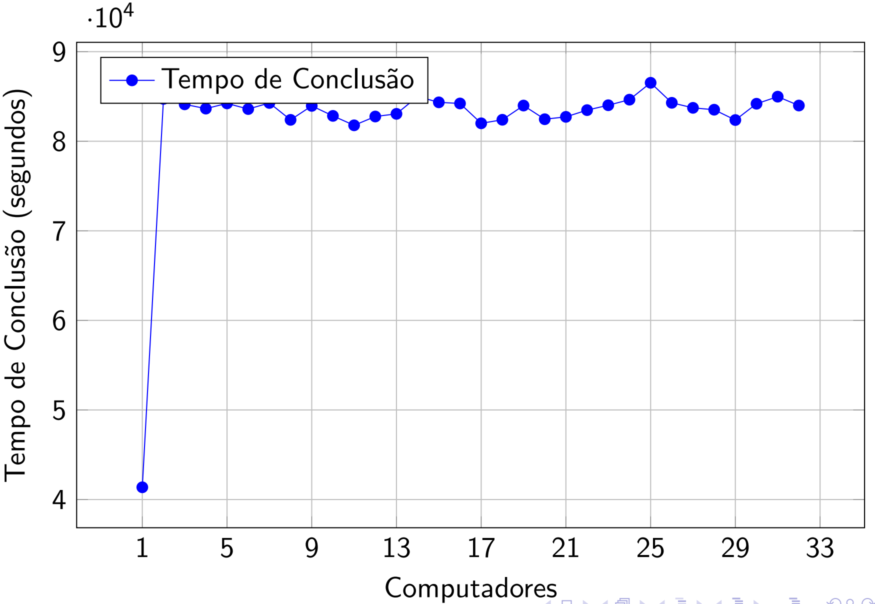


Gráfico 2: Tempo Total de Tarefas por Computador

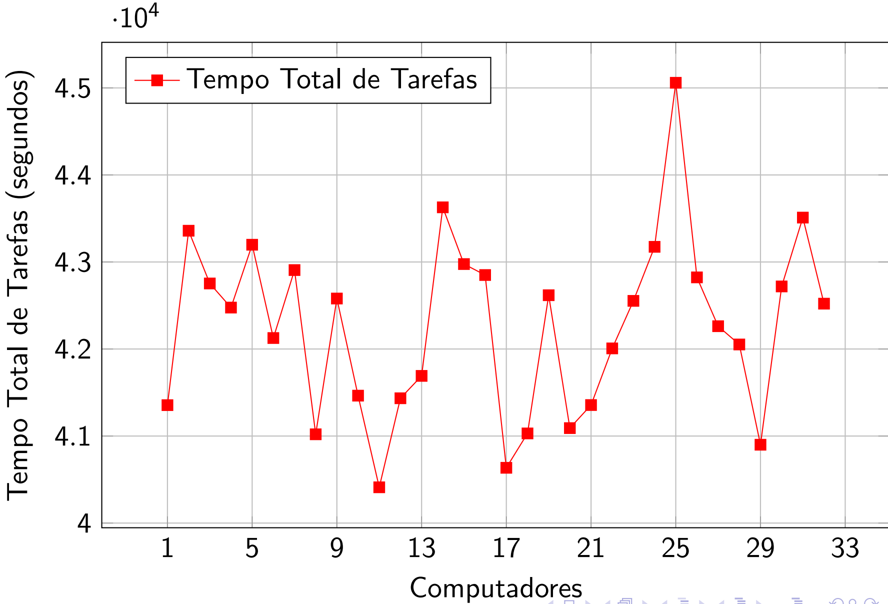
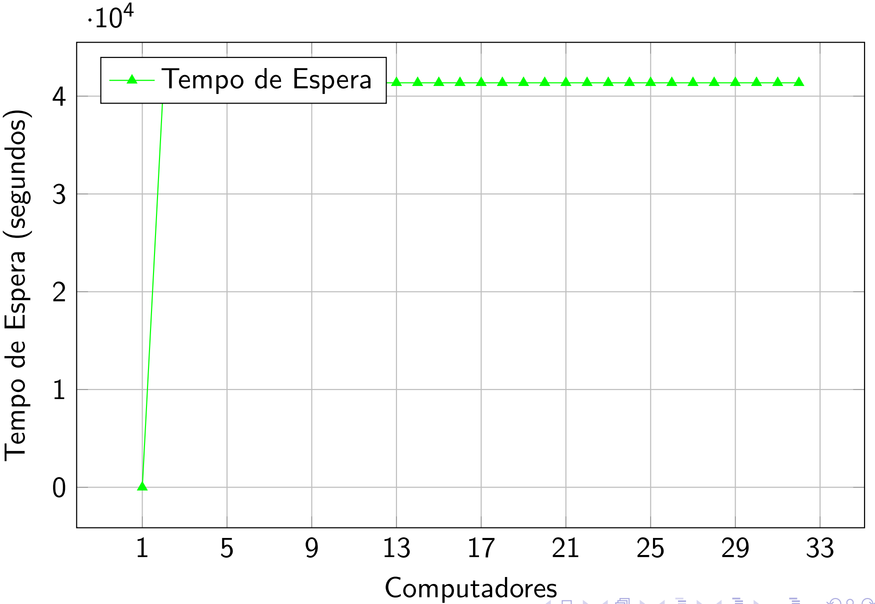


Gráfico 3: Tempo de Espera por Computador



Desafios

- O simulador implementa conceitos de distribuição probabilística para modelar processos de negócio.
- Estima tempos de espera e desempenho com base na ocupação dos recursos.
- Próximos passos:
  - Adicionar analise de fluxos distintos.
  - Implementar métricas de desempenho detalhadas.
  - Validar o modelo de cálculo do Waiting time.