Universidade Federal do Rio de Janeiro

UFRJ

# Trabalho Final - Caching em Redes com Perdas e Atrasos

Avaliação e Desempenho - 2020.2


Aline Freire de Rezende DRE: 116110571

Letícia Tavares da Silva DRE: 117210390

# Sumário

# 1. Introdução

O objetivo deste trabalho é construir um simulador de um sistema de 2 caches em redes (FIFO, LRU, Random e Estáticas) com perdas e atrasos, de acordo com 4 cenários possíveis. São eles: canais sem perdas e sem atrasos, canais com perdas e sem atrasos, e dois canais com perdas e com atrasos. Além disso, deveremos criar cenários adicionais a nosso gosto.

Também devemos efetuar os cálculos utilizando nossos conhecimentos sobre cadeias de Markov, e compararmos com o intervalo de confiança obtido em cada simulação quando possível.

Calcularemos as probabilidades de sucesso, user miss, tempo médio de serviço de requisições, conforme o que for solicitado em cada cenário.

## 1.1. Estados, Eventos e Ações

Definimos que nossos estados são representados pelas variáveis aleatórias X, relativa a uma cache, e Y, relativa à outra; e cada uma delas pode ter os valores {12, 13, 21, 23, 31, 32}, representando os 3 conteúdos definidos. No caso de 4 conteúdos, elas passam a poder assumir os valores {12, 13, 14, 21, 23, 24, 31, 32, 34, 41, 42, 43}.

Nossos eventos são: A = chega uma requisição que está na cache; B = chega uma requisição que não está na cache; TIMEOUT = requisição não é atendida a tempo pelas caches, e é enviada uma requisição a um servidor alternativo.

Nossas ações são definidas da seguinte maneira:

Se ocorrer o evento A, a requisição é atendida e o conteúdo requisitado chega ao usuário. Se ocorrer o evento B, a cache é atendida pelo servidor, e o conteúdo requisitado é alocado nela, para que ela possa também atender a requisição ao usuário. Se chegar o evento TIMEOUT, a requisição é atendida ao usuário por um servidor alternativo.

## 1.2. Linguagem Utilizada

Optamos pela linguagem Python, por estarmos mais familiarizadas com ela. Além disso, ela fornece uma gama de funções pré-existentes para o nosso auxílio, entre elas funções para plotarmos gráficos.

# 2. Análise Estocástica

Utilizamos da própria linguagem de programação para calcular a solução dos cenários 1 e 2 analiticamente. Assumimos que só precisaremos comparar a solução analítica com os dois primeiros cenários para determinar que nossa simulação funciona corretamente. A partir do terceiro cenário, não a utilizaremos.

O sistema de caches do Cenário I e II podem ser modelados por Cadeias de Markov de tempo discreto. Isso ocorre porque independente do estado para o qual queremos ir, apenas importa o estado atual. Afinal, de todas as mudanças que ocorrem nas caches ao longo do tempo, a única que importa para definir para qual novo estado ela transicionará é a mudança que gerou o estado atual, em que estamos agora.

## 2.1. Desenvolvimento dos Cálculos

Para resolução, declararemos as matrizes de transição dos casos FIFO, LRU, Random e Estático, para uma cache, onde cada estado representa os conteúdos alocados na primeira e segunda posição da cache. Temos os conteúdos A, B e C, podendo ser distribuídos em duas posições de cada cache. Já que, em nenhuma circunstância ocorre de haver dois conteúdos iguais em uma mesma cache, então nunca teremos os estados AA, BB ou CC.

Desenvolvemos algumas funções para nos auxiliar no cálculo, as mais importantes delas sendo as que calculavam o objetivo de cada cenário: sucesso_unindo_matrizes(P, depurar = False), referente ao cenário 1, e user_miss(P, p), referente ao cenário 2.

➢ sucesso_unindo_matrizes(P, depurar = False)

Nela, calculamos a probabilidade de sucesso no sistema inteiro, ou seja, o usuário fazer uma requisição de um conteúdo, e ele estar em pelo menos uma das caches. Como a ordem não importa, calculamos da seguinte maneira:

*Sucesso em 1 cache * sucesso em 1 cache + sucesso em 1 cache * falha em 1 cache.*

Nesse cenário, uma falha representa a ausência do conteúdo requisitado na cache em questão.

➢ user_miss(P, p)

Nela, calculamos a probabilidade de user miss no sistema inteiro, ou seja, o usuário fazer uma requisição de um conteúdo, e, ou ambos os canais falharem; ou um canal falhar e o outro não falhar, mas não ter o conteúdo; ou ambos não falharem mas não terem o conteúdo. Para facilitar o cálculo, como se trata de uma probabilidade, resolvemos calcular 1 - "o inverso do que desejamos encontrar". O inverso se trata de ou nenhum dos canais falharem e ambos terem o conteúdo requisitado; nenhum dos canais falharem, mas só um ter o conteúdo; ou um dos canais falhar e o outro não falhar e ter o conteúdo. Calculamos conforme mostramos a seguir:

*1 - ((1 cache não falha * tem) * (1 cache não falha * tem) + (1 cache não falha * tem) * (1 cache não falha * não tem) + 1 cache falha * (1 cache não falha * tem))*

Nesse cenário, uma falha representa a falha do canal da cache em questão.

Além dessas, criamos outras funções:

➢ sucesso_uma_matriz(P, depurar = False):

Nela, chamamos a função para calcular a distribuição estacionária da matriz de transição, e calculamos, por exemplo, P(A) * π(AB), sendo a probabilidade de sair o conteúdo A multiplicado pela distribuição estacionária (a longo prazo) de se estar no estado AB (a cache ter os conteúdos A e B), o que configura um sucesso para uma cache. Utilizamos de laços aninhados para poder garantir que só multiplicaríamos as probabilidades de cada conteúdo pelas distribuições estacionárias que fossem referentes a estados que contivessem esses conteúdos.

➢ forma_tupla(P, depurar = False):

Apenas serve para juntar a lista de distribuição estacionária com a lista referente a cada estado, para facilitar os cálculos na função anterior.

➢ dist_estacionaria(P):

Construímos uma função para calcular a distribuição estacionária a partir de uma sugestão no Stack Overflow.

➢ GeraMatrizTransicao(conteudos, tipo, depurar = False):

Dada uma lista de conteúdos, por exemplo ["A","B","C"], e um tipo, por exemplo FIFO, ela se utiliza da função auxiliar cacheReceivesReq(tipo, cache1,conteudo, depurar = False) da situação de simulação para calcular cada transição, e retornar o conjunto de estados e a matriz de transição propriamente dita para os casos FIFO, LRU e Estática.

## 2.2. Explicação Matemática

Com essas funções, aplicamos os conceitos aprendidos do capítulo 3 do livro "Introduction to Stochastic Processes with R", escrito por Robert P. Dobrow.

Visto que a distribuição estacionária se trata da proporção de transições a longo prazo, a utilizamos para calcular a probabilidade, seguindo a definição 3.2 do livro:

**Stationary Distribution**

Let $X_0, X_1, \ldots$ be a Markov chain with transition matrix $\boldsymbol{P}$. A *stationary distribution* is a probability distribution $\boldsymbol{\pi}$, which satisfies

$$\boldsymbol{\pi} = \boldsymbol{\pi P}. \tag{3.2}$$

That is,

$$\pi_j = \sum_i \pi_i P_{ij}, \text{ for all } j.$$

Ao aplicarmos a definição, conseguiremos as informações relativas à estabilidade das transições de nossa cadeia.

Dessa forma, calculamos as probabilidades de se estar em cada estado, multiplicando a distribuição estacionária π assumindo um papel de distribuição inicial (visto que queremos saber os resultados a longo prazo) pela probabilidade de sair um conteúdo que esteja no estado relativo à distribuição. Para esclarecer, vejamos o seguinte exemplo:

π_ABAB * P(A) representa a probabilidade a longo prazo de se estar no estado ABAB e a probabilidade de sair o conteúdo A. Como existe o conteúdo A na cache, que é o que o estado ABAB representa, essas probabilidades representam um sucesso.

Executando esse cálculo para todos os estados, obteremos o sucesso total do nosso sistema de caches. Dessa forma, esse será o cálculo utilizado para a obtenção da probabilidade de sucesso no cenário I.

# 3. Cenário I – Sem perdas e sem atrasos

O cenário I é composto por duas caches, cada uma com tamanho igual à dois, não há atrasos nos canais de comunicação e é considerado o caso especial onde não há chances de um canal falhar. Dessa forma, a cada requisição feita pelo cliente, apenas é avaliado se as caches possuem ou não o conteúdo requisitado e uma mudança é realizada nos conteúdos da cache de acordo com seu modelo. Nosso propósito nesse cenário é identificar qual a probabilidade de uma requisição ser atendida com sucesso, ou seja, pelo menos uma das duas caches possuir o conteúdo requisitados para cada uma das seguintes composições:

2 caches FIFO; 2 caches LRU; 2 caches Random; 2 caches Estáticas

## 3.1. Simulador

Para simularmos o cenário I, criamos um simulador de eventos de tempo contínuo onde requisições de cada conteúdo chegam numa taxa igual a probabilidade do conteúdo ser requisitado dada e verificamos quantas delas foram atendidas por pelo menos uma das duas caches. Abaixo, temos uma melhor descrição do funcionamento do simulador e da função que o implementa.

### 3.1.1. Descrição do Simulador

O simulador possui um sistema com duas caches onde podemos definir o tamanho delas, o número de conteúdos possíveis de serem requisitados em cada experimento, a probabilidade de requisição de cada conteúdo e os estados iniciais das caches. Inicialmente, as caches são inicializadas com os estados passados, caso não sejam passados, elas são inicializadas de forma aleatória com os conteúdos possíveis, que nesse caso serão representados por números inteiros de 1 ao “numConteudos” dado.

Após isso, agenda-se um evento de uma requisição para cada conteúdo na lista de eventos, esse evento indica qual o tempo em que a requisição irá acontecer e qual conteúdo estará sendo requisitado. O tempo do agendamento é gerado de forma aleatória seguindo a distribuição exponencial tendo como parâmetro a taxa de chegada do conteúdo, que aqui adotamos como 1 * probabilidade do conteúdo ser requisitado. A cada vez que um evento é adicionado, a lista é ordenada em função dos tempos apresentados em cada evento.

Com as caches inicializadas e uma requisição agendada para cada conteúdo, começa-se a realização dos eventos, ou seja, as requisições agendadas começam a ser a realizadas. Em termos de código, inicia-se um loop em cima da lista de eventos, onde em cada iteração, a requisição com menor tempo presente na lista é encaminhada paras as duas caches e para cada cache é realizado o seguinte processo: analisa-se se a cache possui o conteúdo requisitado e assim marca-se se a requisição foi atendida ou não, em ambos os casos a cache muda de estado seguindo o padrão do tipo de cache especificado. Após ser encaminhada para as duas caches, verifica-se a requisição foi atendida por pelo menos uma das caches, se sim, é marcado como sucesso. Após uma requisição ser feita, é agendada uma nova requisição para o mesmo conteúdo onde o tempo do agendamento é gerado de forma aleatória seguindo a distribuição exponencial com parâmetro igual a probabilidade do conteúdo, dessa forma garantimos que cada conteúdo seja requisitado seguindo sua taxa de chegada.

Após o número de requisições pedidas serem feitas, verifica-se o número de sucessos e finalmente calcula-se a probabilidade de sucesso fazendo o seguinte cálculo:

$$n\acute{u}mero\ de\ sucessos / n\acute{u}mero\ de\ requisi\c{c}\tilde{o}es\ realizadas$$

- Parâmetros de entrada para a função do simulador

O simulador do cenário I é dado pela função “simulacaoCenario1()” que tem os seguintes parâmetros de entrada:

* numRequisicoes - Número de requisições que serão realizadas no experimento. Deve ser um número inteiro maior que 0;

* caso - qual é o caso/tipo das caches. Os possíveis valores de entrada são: "FIFO", "LRU", "Random" ou "Estatica";

* numConteudos - número de conteúdos que podem ser requisitados. Deve ser um número inteiro maior que 0;

* tamCache - tamanho das caches. Deve ser um número inteiro maior que 0;

* probabilidades - probabilidade de cada conteúdo ser requisitado. Deve ser passado uma lista de tamanho igual ao número de conteúdos. Por exemplo, se são três conteúdos e possuem probabilidade de requisição uniforme, a lista passada deverá ser [1/3,1/3,1/3];

* depurar - Booleano para indicar se deseja imprimir a depuração do código ou não. O default é o valor "False", onde não se depura;

* cache1_inicial - Estado inicial de uma cache. Se nada for passado, ela irá ser inicializada aleatoriamente;

* cache2_inicial - Estado inicial da outra cache. Se nada for passado, ela irá ser inicializada aleatoriamente.

* cachesDiferentes - Booleano para indicar se deseja que as caches comecem em estados diferentes ou iguais. O default é o valor "True", onde começam com em estados diferentes.

### 3.1.2. Resultados para 3 conteúdos por Simulação

Para obtermos a probabilidade de sucesso, ou seja, a probabilidade de uma requisição ser atendida, sendo 3 conteúdos possíveis de serem requisitados com probabilidade uniforme, calculamos o intervalo de confiança através da função auxiliar "intervaloDeConfianca()" em que se realiza 10 simulações onde em cada simulação são feitas 100000 requisições. Os limites inferiores e superiores do intervalo são então obtidos da seguinte maneira:

$$limite\ inferior = \frac{media\ dos\ resultados\ das\ simulações - 1.96 * desvio\ padrão\ dos\ resultados\ das\ simulações}{\sqrt[2]{número\ de\ simulações}}$$

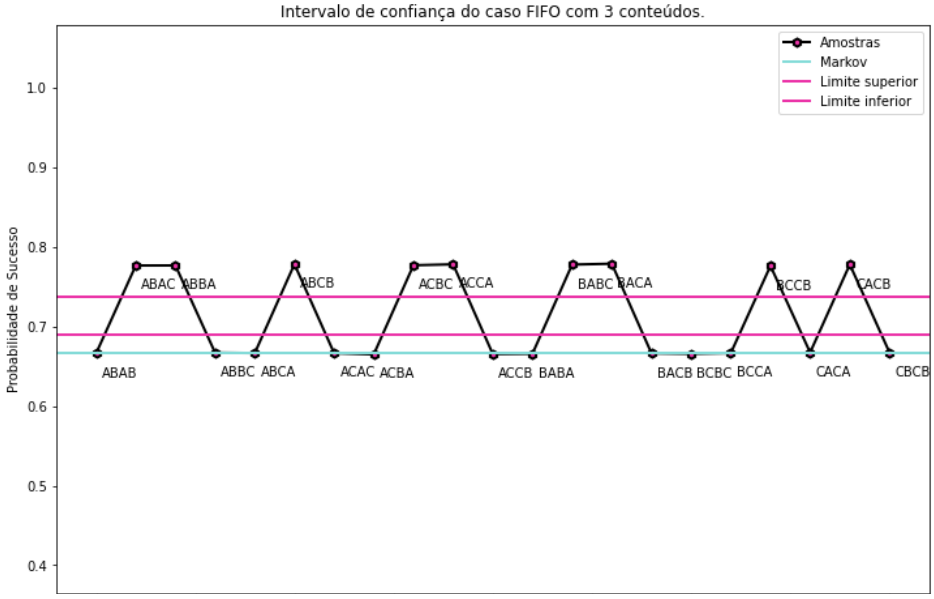
$$limite\ superior = \frac{media\ dos\ resultados\ das\ simulações + 1.96 * desvio\ padrão\ dos\ resultados\ das\ simulações}{\sqrt[2]{número\ de\ simulações}}$$

- Caso FIFO

Para o caso em que as duas caches são do tipo FIFO obtivemos o seguinte intervalo de confiança:

$$[0.7218442078798188, 0.7592897921201813]$$

Vemos que é um grande intervalo, não tão preciso. Acompanhando as execuções de cada simulação, verificamos que isso acontece porque a probabilidade varia entre 0.66 e 0.77 dependendo do estado inicial das caches. Alguns estados iniciais específicos sempre vão fornecer 66% de chances de atender uma requisição, enquanto que outros vão fornecer 77%. Podemos ver a probabilidade de sucesso para cada estado inicial possível na imagem abaixo.



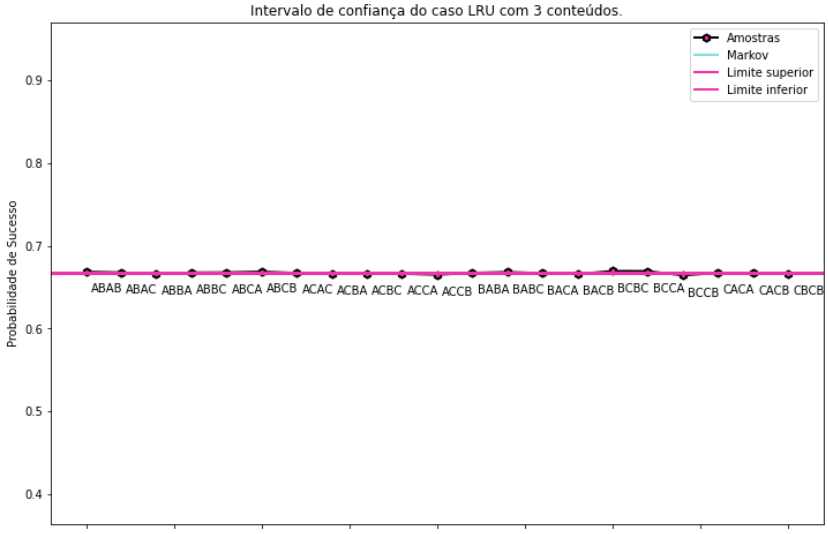

Por mais que pareça que são 36 estados possíveis, já que possuímos 6 estados iniciais possíveis para cada cache. Verificamos que são apenas 21, pois a ordem das caches não importam. Por exemplo, o estado ABCB é o mesmo que CBAB, pois em ambos uma cache possui os conteúdos AB enquanto a outra possui os conteúdos CB sem importar qual cache é qual.

- Caso LRU

Para o caso em que as duas caches são do tipo LRU obtivemos o seguinte intervalo de confiança:

$$[0.6659710487891771,\ 0.6675989512108228]$$

Vemos que no caso LRU obtemos um intervalo menor e mais específico, onde temos que a probabilidade de uma requisição ser atendida é aproximadamente $\frac{2}{3} \cong 66\%$. O intervalo pequeno se deve ao fato de que a independente do estado inicial das caches, a probabilidade de sucesso é sempre $\cong 66\%$, como podemos ver na imagem abaixo.



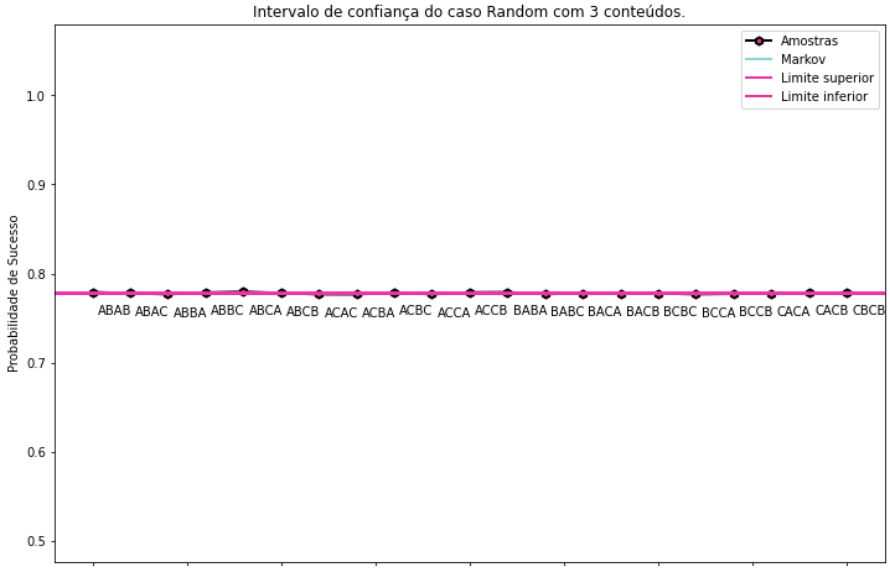

Vemos então que no caso LRU também são 21 estados, pelo mesmo motivo apresentado na FIFO e independente do estado inicial, a probabilidade de sucesso obtida é igual à ⅔.

➢ Caso Random

Para o caso em que as duas caches são do tipo Random obtivemos o seguinte intervalo de confiança:

$$[0.7768191892942723,\ 0.7786568107057278]$$

Podemos observar que no caso Random também obtemos um intervalo menor e mais específico, onde temos que a probabilidade de uma requisição ser atendida é aproximadamente $\frac{7}{9} \cong 77\%$. O intervalo pequeno novamente se deve ao fato de que independente do estado inicial das caches, a probabilidade de sucesso é sempre $\cong 77\%$, como apresentado na imagem abaixo.



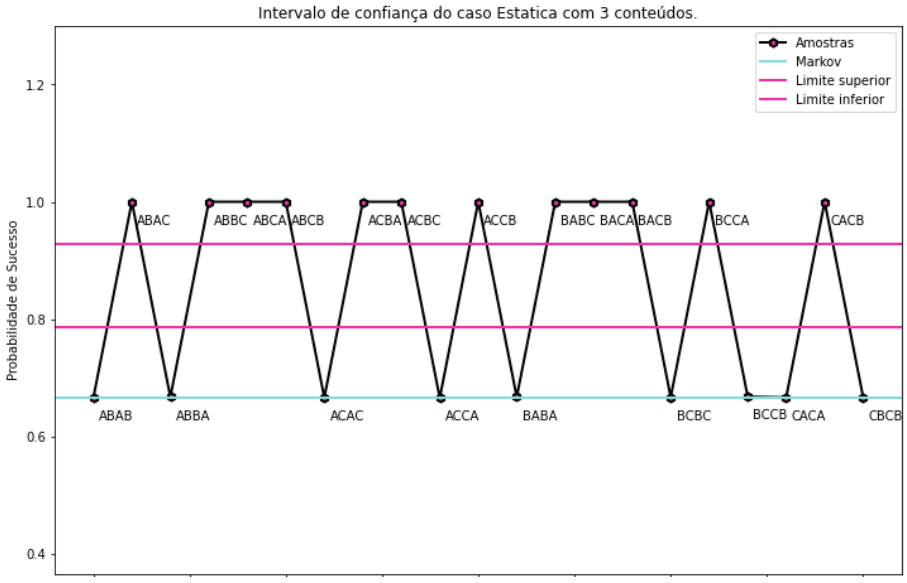
➢ Caso Estática

Para o caso em que as duas caches são do tipo Estática obtivemos o seguinte intervalo de confiança:

$[0.666144805020255,\ 0.668409194979745] \rightarrow$*caso as caches comecem com os mesmos conteúdos*

$[1.0,\ 1.0] \rightarrow$*caso as caches comecem com conteúdos diferentes*

No caso estática temos que a probabilidade de sucesso irá depender dos estados iniciais das caches, assim como no caso FIFO. Se as caches começam com os mesmos conteúdos, a probabilidade de uma requisição ser atendida é aproximadamente $\frac{2}{3} \cong 66\%$, já que sempre iremos ter 2 conteúdos dos 3 três possíveis de serem requisitados com probabilidade uniforme. Já se as caches começam com conteúdos diferentes, a probabilidade será de 100% de sucesso, pois é garantido que os conteúdos são atendidos por pelo menos uma das caches. Vemos a probabilidade para cada estado inicial na imagem abaixo onde podemos notar o que dito antes, os estados iniciais que apresentam probabilidade $\frac{2}{3} \cong 66\%$ são os estados onde as caches começam com os mesmos conteúdos.




## 3.1.3. Resultados para 3 conteúdos por Markov

➢ FIFO

Os estados possíveis para uma cache deste tipo são AB, AC, BA, BC, CA e CB. A matriz de transição para uma cache é dada por::

$$AB\ AC\ BA\ BC\ CA\ CB$$

$$P = AB\ AC\ BA\ BC\ CA\ CB\ [2/3\ 0\ 0\ 0\ 1/3\ 0\ 0\ 2/3\ 1/3\ 0\ 0\ 0\ 0\ 0\ 2/3\ 0\ 0\ 1/3\ 1/3\ 0\ 0\ 2/3\ 0\ 0\ 0\ 0\ 0\ 1/3\ 2/3\ 0\ 0\ 1/3\ 0\ 0\ 0$$

Nela, enxergamos, por exemplo, que a transição de AB para AB tem probabilidade de $\frac{2}{3}$ e de AB para CA tem probabilidade de $\frac{1}{3}$. As probabilidades de transição para quaisquer outros estados são nulas. Isso ocorre porque, ao se requisitar conteúdos que já estejam nela, nada muda; e ao se requisitar um conteúdo que não esteja, ele entra em sua primeira posição, o conteúdo que estava na primeira posição avança para a segunda, e o que estava na segunda sai da mesma.

Ao calcularmos a distribuição estacionária, obtivemos:

$$[\pi_{AB}, \pi_{AC}, \pi_{BA}, \pi_{BC}, \pi_{CA}, \pi_{CB}] = [\tfrac{1}{3}, 0, 0, \tfrac{1}{3}, \tfrac{1}{3}, 0]$$

Assim, obtemos que a probabilidade de sucesso para uma cache é igual á:

$$P(S) = \pi_{AB} * (P(A) + P(B)) + \pi_{BC} * (P(B) + P(C)) + \pi_{CA} * (P(C) + P(A))$$

$$P(S) = \tfrac{1}{3} * \left(\tfrac{2}{3}\right) + \tfrac{1}{3} * \left(\tfrac{2}{3}\right) + \tfrac{1}{3} * \left(\tfrac{2}{3}\right) = \tfrac{2}{3}$$

Assim, a probabilidade de sucesso com as duas caches será:

*Sucesso em 1 cache * sucesso em 1 cache + sucesso em 1 cache * falha em 1 cache.*

$$\tfrac{2}{3} * \tfrac{2}{3} + \tfrac{2}{3} * \tfrac{1}{3} = \tfrac{2}{3} = 0.666 = 66\%$$

Contudo, temos que esse resultado não se encontra no intervalo de confiança obtido na simulação, isso ocorre devido a distribuição estacionária depender do estado inicial. Assim, elaboramos a matriz de transição para as 2 caches juntas que pode ser vista abaixo para mostrarmos essa dependência do estado inicial observada pela simulação. Os estados possíveis para um par de caches deste tipo são ABAB, ABAC, ABBA, ABBC, ABCA, ABCB, ACAC, ACBA, ACBC, ACCA, ACCB, BABA, BABC, BACA, BACB, BCBC, BCCA, BCCB, CACA, CACB e CBCB.

| | ABAB | ABAC | ABBA | ABBC | ABCA | ABCB | ACAC | ACBA | ACBC | ACCA | ACCB | BABA | BABC | BACA | BACB | BCBC | BCCA | BCCB | CACA | CACB | CBCB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABAB | 0,667 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ABAC | 0 | 0,333 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ABBA | 0 | 0 | 0,667 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ABBC | 0,333 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 |
| ABCA | 0 | 0 | 0 | 0,333 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ABCB | 0 | 0,333 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ACAC | 0 | 0 | 0 | 0 | 0 | 0 | 0,667 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACBA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACBC | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACCA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,667 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACCB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 |
| BABA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,667 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BABC | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 |
| BACA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| BACB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BCBC | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,667 | 0 | 0 | 0 | 0 | 0 |
| BCCA | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0,333 | 0 | 0 | 0 | 0 |
| BCCB | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,667 | 0 | 0 | 0 |
| CACA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,667 | 0 | 0 |
| CACB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0,333 | 0 |
| CBCB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,667 |

Em seguida, elevamos a matriz a uma potência alta, no caso 100, para verificarmos a distribuição estacionária para cada estado inicial. Podemos observar matriz $P^{100}$ abaixo.

| | ABAB | ABAC | ABBA | ABBC | ABCA | ABCB | ACAC | ACBA | ACBC | ACCA | ACCB | BABA | BABC | BACA | BACB | BCBC | BCCA | BCCB | CACA | CACB | CBCB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABAB | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| ABAC | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| ABBA | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| ABBC | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| ABCA | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| ABCB | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| ACAC | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| ACBA | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| ACBC | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| ACCA | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| ACCB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BABA | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BABC | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| BACA | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| BACB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BCBC | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| BCCA | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| BCCB | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| CACA | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| CACB | 0 | 0,111 | 0,222 | 0 | 0 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0 | 0,111 | 0 | 0 | 0 | 0 | 0,222 | 0 | 0,111 | 0 |
| CBCB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |

Como mostrado na imagem acima, as linhas da matriz que simula 100 passos não são iguais, confirmando assim que a distribuição estacionária realmente depende do estado inicial, onde algumas irão fornecer a probabilidade de sucesso igual à $\frac{2}{3} \cong 66\%$ enquanto outras fornecerão $\frac{7}{9} \cong 77\%$.

➢ LRU

Os estados possíveis para uma cache deste tipo também são AB, AC, BA, BC, CA e CB. A matriz de transição para uma cache é dada por:

$$P = \begin{matrix} & AB & AC & BA & BC & CA & CB \\ AB \\ AC \\ BA \\ BC \\ CA \\ CB \end{matrix} \begin{bmatrix} 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 0 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 1/3 & 0 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Nela, enxergamos, por exemplo, que as transições de AB para AB, de AB para BA e de AB para CA todas têm probabilidade de ⅓. As probabilidades de transição para quaisquer outros estados são nulas. Isso ocorre porque, ao se requisitar um conteúdo que já esteja nela, ele é trazido para a primeira posição, e o outro shifta para a segunda posição; e ao se requisitar um conteúdo que não esteja, ele entra em sua primeira posição, o conteúdo que estava na primeira posição avança para a segunda, e o que estava na segunda sai da mesma, assim como no caso FIFO. Ao calcularmos a distribuição estacionária, obtivemos:

$$[\pi_{AB}, \pi_{AC}, \pi_{BA}, \pi_{BC}, \pi_{CA}, \pi_{CB}] = [\tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}, \tfrac{1}{6}]$$

Assim, obtemos que a probabilidade de sucesso para uma cache é igual á:

$$P(S) = \pi_{AB} * (P(A) + P(B)) + \pi_{AC} * (P(A) + P(C)) + \pi_{BA} * (P(B) + P(A)) + \pi_{BC} * (P(B) + P(C)) + \pi_{CA} * (P(C)$$

$$P(S) = \tfrac{1}{6} * (\tfrac{2}{3}) + \tfrac{1}{6} * (\tfrac{2}{3}) + \tfrac{1}{6} * (\tfrac{2}{3}) + \tfrac{1}{6} * (\tfrac{2}{3}) + \tfrac{1}{6} * (\tfrac{2}{3}) + \tfrac{1}{6} * (\tfrac{2}{3}) = \tfrac{12}{18} = \tfrac{2}{3}$$

Assim, a probabilidade de sucesso com as duas caches será:

*Sucesso em 1 cache * sucesso em 1 cache + sucesso em 1 cache * falha em 1 cache.*

$$\tfrac{2}{3} * \tfrac{2}{3} + \tfrac{2}{3} * \tfrac{1}{3} = \tfrac{2}{3} = 0.666 = 66\%$$

Esse resultado se encontra no intervalo de confiança obtido na simulação. Elaboramos também a matriz de transição para as 2 caches juntas que pode ser vista abaixo para comprovarmos a não dependência da distribuição estacionária pelo estado inicial observada na simulação. Os estados possíveis para um par de caches deste tipo também são ABAB, ABAC, ABBA, ABBC, ABCA, ABCB, ACAC, ACBA, ACBC, ACCA, ACCB, BABA, BABC, BACA, BACB, BCBC, BCCA, BCCB, CACA, CACB e CBCB.

| | ABAB | ABAC | ABBA | ABBC | ABCA | ABCB | ACAC | ACBA | ACBC | ACCA | ACCB | BABA | BABC | BACA | BACB | BCBC | BCCA | BCCB | CACA | CACB | CBCB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABAB | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ABAC | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ABBA | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ABBC | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ABCA | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ABCB | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ACAC | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ACBA | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ACBC | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| ACCA | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 |
| ACCB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| BABA | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BABC | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BACA | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 |
| BACB | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 |
| BCBC | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0,333 |
| BCCA | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 |
| BCCB | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0,333 |
| CACA | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0,333 | 0 | 0 |
| CACB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0,333 | 0 |
| CBCB | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,333 | 0 | 0 | 0 | 0 | 0,333 |

Em seguida, elevamos a matriz a uma potência alta, no caso 100, para verificarmos a distribuição estacionária para cada estado inicial. Podemos observar matriz $P^{100}$ abaixo.

| | ABAB | ABAC | ABBA | ABBC | ABCA | ABCB | ACAC | ACBA | ACBC | ACCA | ACCB | BABA | BABC | BACA | BACB | BCBC | BCCA | BCCB | CACA | CACB | CBCB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABAB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ABAC | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ABBA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ABBC | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ABCA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ABCB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ACAC | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ACBA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ACBC | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ACCA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| ACCB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BABA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BABC | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BACA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BACB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BCBC | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BCCA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| BCCB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| CACA | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| CACB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| CBCB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |
| CBCB | 0,167 | 0 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0 | 0,167 | 0 | 0 | 0,167 | 0 | 0,167 |

Como as linhas da matriz que simula 100 passos são iguais, confirmando assim que a distribuição estacionária de fato não depende do estado inicial, onde todos fornece a probabilidade de sucesso igual à $\frac{2}{3} \cong 66\%$.

➢ Random

Como para esse caso a matriz de transição para as 2 caches é menor, optamos por resolver logo por ela. Abaixo então podemos ver a matriz de transição de 2 caches para o tipo Random.

| | ABAB | ABBC | ABCA | BCBC | BCCA | CACA |
|---|---|---|---|---|---|---|
| ABAB | 0,667 | 0 | 0 | 0,083 | 0,167 | 0,083 |
| ABBC | 0,167 | 0,333 | 0,167 | 0,167 | 0,167 | 0 |
| ABCA | 0,167 | 0,167 | 0,333 | 0 | 0,167 | 0,167 |
| BCBC | 0,083 | 0 | 0,167 | 0,667 | 0 | 0,083 |
| BCCA | 0 | 0,167 | 0,167 | 0,167 | 0,333 | 0,167 |
| CACA | 0,083 | 0 | 0,167 | 0,083 | 0 | 0,667 |

Nela, enxergamos, por exemplo, que a transição de ABAB para BCCA tem probabilidade de $\frac{1}{6}$ e de ABAB para ABAB $\frac{2}{3}$. Isso ocorre porque, ao se requisitar um conteúdo que já esteja nela, nada muda; e ao se requisitar um conteúdo que não esteja, ele entra no lugar de um outro aleatoriamente selecionado. Assim, a ordem dos conteúdos nos estados não importa.

Em seguida, elevamos a matriz á uma potência alta, no caso 100, para verificarmos a distribuição estacionária para cada estado inicial. Podemos observar matriz $P^{100}$ abaixo.

| | ABAB | ABBC | ABCA | BCBC | BCCA | CACA |
|---|---|---|---|---|---|---|
| ABAB | 0,222 | 0,067 | 0,156 | 0,204 | 0,111 | 0,24 |
| ABBC | 0,222 | 0,067 | 0,156 | 0,204 | 0,111 | 0,24 |
| ABCA | 0,222 | 0,067 | 0,156 | 0,204 | 0,111 | 0,24 |
| BCBC | 0,222 | 0,067 | 0,156 | 0,204 | 0,111 | 0,24 |
| BCCA | 0,222 | 0,067 | 0,156 | 0,204 | 0,111 | 0,24 |
| CACA | 0,222 | 0,067 | 0,156 | 0,204 | 0,111 | 0,24 |

Como as linhas da matriz que simula 100 passos são iguais, confirmamos assim que a distribuição estacionária de fato não depende do estado inicial como no caso LRU, onde todos fornece a probabilidade de sucesso igual à $\frac{7}{9} \cong 77\%$ como calculado abaixo:.

$$[\pi_{ABAB}, \pi_{ABBC}, \pi_{ABCA}, \pi_{BCBC}, \pi_{BCCA}, \pi_{CACA}] = [0.222,\ 0.067,\ 0.156,\ 0.204,\ 0.111,\ 0.24]$$

Assim, a probabilidade de sucesso com as duas caches será:

$$P(S) = \pi_{ABAB} * (P(A) + P(B)) + \pi_{ABBC} * (P(A) + P(B) + P(C)) + \pi_{ABCA} * (P(A) + P(C) + P(B)) + \pi_{BCBC} * (P(C)$$

$$P(S) = 0.222 * \frac{2}{3} + 0.067 * 1 + 0.156 * 1 + 0.204 * \frac{2}{3} + 0.111 * 1 + 0.24 * \frac{2}{3} = 0.77 = 77\%$$

➢ Estática

Como para esse caso a matriz de transição é fácil de elaborar por ser a matriz identidade, já que no caso das caches estáticas nunca saímos do estado inicial, montamos a matriz de transição para

as duas caches juntas onde estados possíveis para um par de caches deste tipo também são ABAB, ABAC, ABBA, ABBC, ABCA, ABCB, ACAC, ACBA, ACBC, ACCA, ACCB, BABA, BABC, BACA, BACB, BCBC, BCCA, BCCB, CACA, CACB e CBCB. Ela pode ser vista abaixo:

| | ABAB | ABAC | ABBA | ABBC | ABCA | ABCB | ACAC | ACBA | ACBC | ACCA | ACCB | BABA | BABC | BACA | BACB | BCBC | BCCA | BCCB | CACA | CACB | CBCB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ABAB | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ABAC | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ABBA | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ABBC | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ABCA | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ABCB | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACAC | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACBA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACBC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACCA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ACCB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BABA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BABC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BACA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BACB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCBC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BCCA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| BCCB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| CACA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| CACB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| CBCB | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Nela, enxergamos, por exemplo, que a transição de ABAB para ABAB tem probabilidade de 1. As probabilidades de transição para quaisquer outros estados são nulas. Isso ocorre porque, no caso de uma cache estática, ela é populada com os conteúdos mais populares, e a partir de então nunca mais mudará. Como a matriz identidade elevada a qualquer potência é a própria matriz identidade, temos que matriz $P^{100}$ também é dada pela matriz identidade onde as linhas não são iguais. Assim, como encontrado na simulação, a distribuição estacionária depende do estado inicial, onde alguns irão fornecer a probabilidade de sucesso igual à $\frac{2}{3} \cong 66\%$ enquanto outras fornecerão $1 = 100\%$.

### 3.1.4. Resultados para 4 conteúdos por Simulação

Para obtermos a probabilidade de sucesso, ou seja, a probabilidade de uma requisição ser atendida, sendo 4 conteúdos possíveis de serem requisitados com probabilidade uniforme, calculamos o intervalo de confiança através da função auxiliar "intervaloDeConfianca()" em que se realiza 10 simulações onde em cada simulação são feitas 100000 requisições. Os limites inferiores e superiores do intervalo são então obtidos da seguinte maneira:

$$\textit{limite inferior} = \frac{\textit{media dos resultados das simulações} - 1.96 * \textit{desvio padrão dos resultados das simulações}}{\sqrt[2]{\textit{número de simulações}}}$$

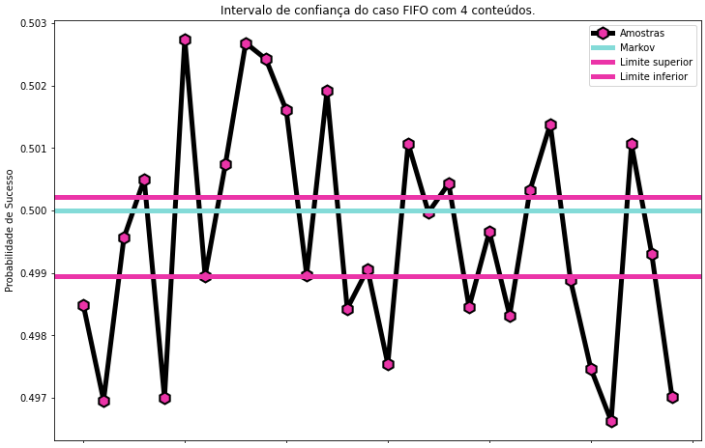
$$\textit{limite superior} = \frac{\textit{media dos resultados das simulações} + 1.96 * \textit{desvio padrão dos resultados das simulações}}{\sqrt[2]{\textit{número de simulações}}}$$

- Caso FIFO

Para o caso em que as duas caches são do tipo FIFO obtivemos o seguinte intervalo de confiança:

[0.4989537339736462, 0.5002129326930206]

Vemos que para 4 conteúdos, a situação para o caso FIFO muda, o intervalo passa a ser preciso. Acompanhando as execuções de cada simulação, verificamos que a probabilidade de sucesso para qualquer estado inicial é aproximadamente 50%. Podemos ver abaixo a probabilidade de sucesso para cada 30 simulações realizadas onde em cada uma são feitas 100000 requisições:

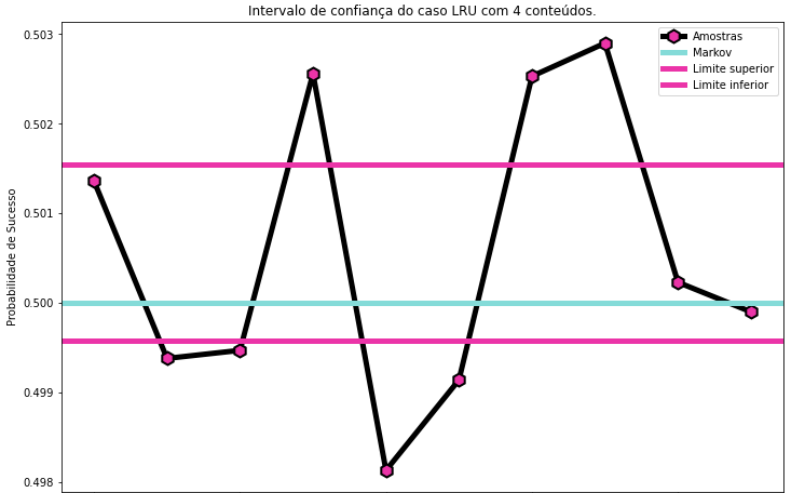



➢ Caso LRU

Para o caso em que as duas caches são do tipo LRU obtivemos o seguinte intervalo de confiança:

$$[0.4995791614787336,\ 0.5015408385212664]$$

Vemos que no caso LRU novamente obtemos um intervalo menor e mais específico, onde temos que a probabilidade de uma requisição ser atendida é aproximadamente $\frac{1}{2} = 50\%$. O intervalo pequeno se deve ao fato de que a independente do estado inicial das caches, a probabilidade de sucesso é sempre $\cong 50\%$, como podemos ver na imagem abaixo que apresenta probabilidade de sucesso para cada simulação realizada no cálculo do intervalo de confiança:




➢ Caso Random

Para o caso em que as duas caches são do tipo Random obtivemos o seguinte intervalo de confiança:

$$[0.6092824298672316,\ 0.6108615701327685]$$

Podemos observar que no caso Random obtemos novamente um intervalo menor e mais específico, onde temos que a probabilidade de uma requisição ser atendida é aproximadamente $\frac{6}{10} \cong 60\%$. O intervalo pequeno se deve ao fato de que a independente do estado inicial das caches, a probabilidade de sucesso é sempre $\cong 60\%$, como apresentado na imagem abaixo que apresenta probabilidade de sucesso para cada simulação realizada no cálculo do intervalo de confiança:


➢ Caso Estática

Para o caso em que as duas caches são do tipo Estática obtivemos o seguinte intervalo de confiança:

$[0.49871813993220265,\ 0.5004798600677974] \rightarrow$ *caso as caches comecem com os mesmos conteúdos*
$[0.7494378116151015,\ 0.7510041883848986] \rightarrow$ *caso as caches comecem com conteúdos diferentes*

No caso estática temos que a probabilidade de sucesso continua dependendo dos estados iniciais das caches, pois, se as caches começam com os mesmos conteúdos, a probabilidade de uma requisição ser atendida é aproximadamente $\cong 50\%$, já que sempre iremos ter 2 conteúdos dos 4 três possíveis de serem requisitados com probabilidade uniforme. Já se as caches começam com conteúdos diferentes, alguns irão fornecer a probabilidade de 75%, pois têm 3 conteúdos dos 4 possíveis como o estado ABBC enquanto outros irão fornecer 100%, como o estado ABCD.

## 3.1.5. Análise do Cenário I

**Qual tipo de cache é melhor?**

Sempre preferiremos caches cujos conteúdos sejam diferentes. Exemplo: o par de caches [A,B] e [A,C]. Dessa forma, uma requisição de qualquer tipo (A, B ou C) sempre será atendida, já que temos todas as possibilidades de conteúdos requisitados guardados nelas.

Percebemos que nos casos FIFO e LRU, ainda que elas comecem diferentes, a partir do momento que ficarem iguais, nunca mais ficarão diferentes. No caso estático, só teremos caches com conteúdos diferentes se elas começarem diferentes, já que nunca mudarão ao longo das requisições (há a chance de começarem diferentes pois a popularidade de conteúdo neste cenário é determinada pela distribuição da probabilidade, que neste caso é uniforme).

Assim, concluímos que a Random é a melhor para a nossa busca do sucesso. Em qualquer momento a cache pode ficar diferente, ainda que em algum momento elas fiquem iguais, pois o novo conteúdo entrará em uma posição aleatória de cada cache. Assim teremos uma maior probabilidade de a requisição ser atendida.

Contudo, vemos que a FIFO apresenta estados iniciais diferentes que nunca conseguem chegar em estados iguais (vemos isso pois a cadeia FIFO é redutível e tem 3 classes). Assim, esses casos também apresentam o melhor resultado no caso de termos um conteúdo a mais que o tamanho das caches. Assim, para 3 conteúdos no cenário I, esses casos da FIFO seriam tão bons quanto as caches Random, mas a partir de 4 conteúdos elas perdem essa vantagem. Então é mais vantajoso escolher a Random,

**Irredutíveis ou redutíveis?**

Caso N = 3.

Nomearemos aqui os estados de 1 a 21, seguindo a ordem dos estados das matrizes, para facilitar a escrita e a explicação. Além disso, todas as matrizes de transição foram expostas na seção de declarações da solução analítica.

As cadeias FIFO são todas redutíveis:

- FIFO com uma cache: caminhos:
  - $1 \longrightarrow 5 \longrightarrow 4 \longrightarrow 1$;
  - $2 \longrightarrow 3 \longrightarrow 6 \longrightarrow 2$.

Podemos também fazer essa análise pelo grafo:

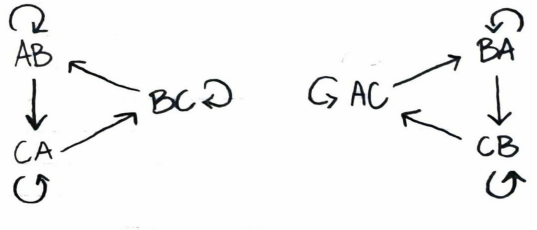



**Figura 1. Grafo da matriz de transição do caso FIFO representando uma única cache.**

Temos duas classes recorrentes, completamente desconexas.

- FIFO com duas caches: caminhos:

- 1⟶19⟶16⟶1;
- 2⟶3⟶20⟶10⟶13⟶18⟶2;
- etc.

Podemos também fazer essa análise pelo grafo:

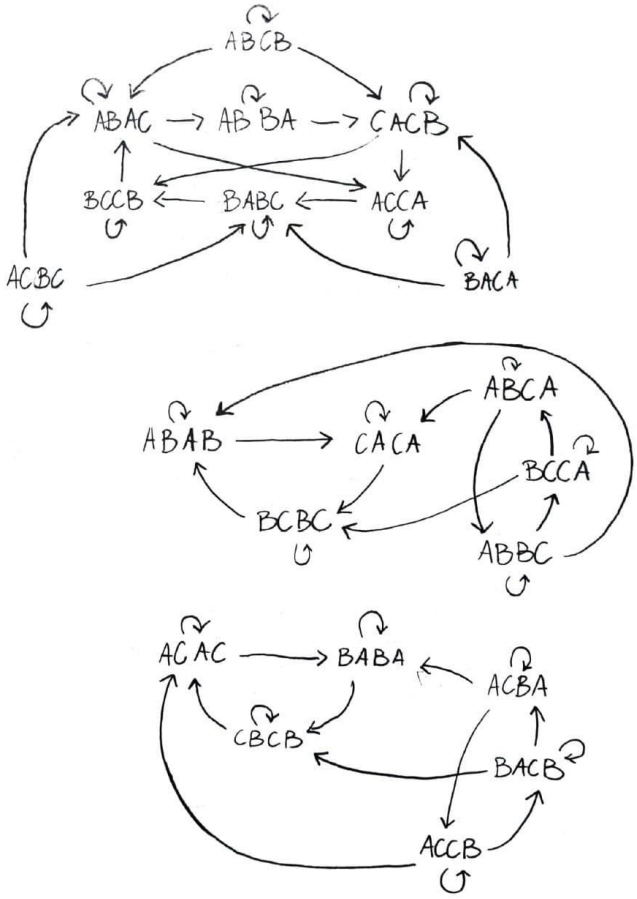



**Figura 2. Grafo da matriz de transição do caso FIFO representando duas caches.**

Esse grafo é desconexo, e temos 3 conjuntos conexos. Neles, ainda percebemos que não é possível chegar em alguns estados, como por exemplo o estado ACBC. No primeiro conjunto conexo, enxergamos que é impossível ter as duas caches com os mesmos conteúdos. No segundo e no terceiro conjuntos, percebemos que se começarmos de algum estado cujas caches são diferentes, é possível que eles transicionem para estados cujas caches são iguais, pois esses primeiros estados são transientes. A partir do momento que elas se tornam iguais, conseguimos enxergar que elas nunca mais ficarão diferentes. As classes dos estados cujas caches são iguais são recorrentes.

As cadeias LRU:

➢ LRU com uma cache é irredutível: caminhos:
  - 1⟶3⟶6⟶2⟶5-⟶4⟶1.

Completa! Dá para chegar em todos os estados começando de todos os estados.

➢ LRU com duas caches é redutível: caminhos:
  - 1⟶12⟶21⟶7⟶19⟶16-⟶1;

- 2⟶12⟶1⟶19⟶7⟶19⟶16⟶21⟶12⟶1;
- etc.

Analogamente ao caso FIFO, vemos que não é possível chegar no 2 pelo 1. É interessante ressaltar que o oposto é possível.

As cadeias Random são todas irredutíveis:

➢ Random (uma cache): caminho:
- 1⟶2⟶3⟶1.

Completa! Dá para chegar em todos os estados começando de todos os estados.

➢ Random (duas caches): caminho:
- 1⟶4⟶3⟶2⟶5⟶6⟶1.

Completa! Dá para chegar em todos os estados começando de todos os estados.

Todas as cadeias Estáticas são redutíveis. Não há nenhuma comunicação entre os estados. Independente de qual se começar, não é possível mudar de estado.

**O desempenho depende do estado inicial?**

Como afirmamos em tópicos anteriores, o desempenho depende para alguns tipos de caches. Conforme discutimos na primeira pergunta, se as caches começarem iguais, só haverá casos de caches com conteúdos diferentes, sendo 77% de chance de sucesso a cada requisição, no caso das caches Random.

Se elas começarem diferentes, antes da convergência da FIFO e LRU, também haverá essa chance de 77% de ter qualquer conteúdo requisitado salvo na cache.

E as Estáticas permanecerão diferentes até o fim, já que nunca mudam, configurando 100% de chance de ter qualquer conteúdo requisitado salvo na cache sempre.

Uma explicação é a abordada anteriormente em que ao elevarmos uma matriz de transição a um número muito grande, para representar a passagem de vários passos, atinge-se uma distribuição estacionária nessa matriz, onde todas as linhas são números iguais. Com ela, temos que a probabilidade de se terminar em um mesmo estado é a mesma. Ela também é a distribuição limitante.

Mas há exceções. Em casos que, elevada a um número muito alto, a matriz não fica com as linhas iguais. Isso quer dizer que há mais de uma distribuição estacionária, e que a probabilidade de se estar em algum estado necessariamente dependerá de qual é o estado inicial.

É o que acontece com as matrizes de transição FIFO e Estática.

# 4. Cenário II – Com perdas e sem atrasos

O cenário II também é composto por duas caches, cada uma com tamanho igual à dois, não há atrasos nos canais de comunicação, contudo há a probabilidade de um canal falhar. Dessa forma, a cada requisição feita pelo cliente, primeiramente é avaliado se o canal funcionou e só assim é avaliado se a cache possui ou não o conteúdo requisitado. Nosso propósito nesse cenário é identificar qual a

probabilidade de ocorrer um user miss que em outras palavras é a probabilidade de uma requisição não ser atendida, para cada uma das seguintes composições:

2 caches FIFO; 2 caches LRU; 2 caches Random; 2 caches Estáticas

## 4.1. Simulador

Para simularmos o cenário II, criamos um simulador de eventos de tempo contínuo muito semelhante ou simulador do cenário I onde requisições de cada conteúdo chegam a uma taxa igual a probabilidade do conteúdo ser requisitado dada e verificamos quantas delas foram atendidas por pelo menos uma das duas caches. A única diferença nesse caso é que calcula-se se o canal falhou ou não, com probabilidade igual a p = 0.9 de funcionar, antes encaminhar para a cache, Abaixo, temos uma melhor descrição do funcionamento do simulador e da função que o implementa.

### 4.1.1. Descrição do Simulador

Como o cenário II se assemelha muito ao cenário I, a descrição do simulador é bastante semelhante. Temos então que inicialmente, as caches são inicializadas com os estados passados, caso não sejam passados, elas são inicializadas de forma aleatória com os conteúdos possíveis, que nesse caso serão representados por números inteiros de 1 ao numConteudos dado.

Após isso, agenda-se um evento de uma requisição para cada conteúdo na lista de eventos, esse evento indica qual o tempo em que a requisição irá acontecer e qual conteúdo estará sendo requisitado. O tempo do agendamento é gerado de forma aleatória seguindo a distribuição exponencial tendo como parâmetro a taxa do conteúdo e a cada vez que um evento é adicionado, a lista é ordenada em função dos tempos apresentados em cada evento.

Com as caches inicializadas e uma requisição agendada para cada conteúdo, começa-se a realização dos eventos, ou seja, as requisições agendadas começam a ser realizadas. Em termos de código, inicia-se um loop em cima da lista de eventos, onde em cada iteração, a requisição com menor tempo presente na lista é encaminhada para as duas caches. Neste ponto que o cenário II se diferencia do cenário I, neste caso há a probabilidade do canal que leva a requisição até a cache falhar, e com isso não tem a possibilidade da requisição ser atendida pela cache que por conta disso não mudará de estado.

Então para cada cache é realizado o seguinte processo: tenta-se encaminhar a requisição para cache, para isso gera-se um número entre 0 e 1, se o número for menor ou igual ao parâmetro $p$ (que por default é $0.9$, o que implica que o canal tem 90% de chance de funcionar), o canal funciona e a requisição alcança a cache e assim analisa-se se a cache possui o conteúdo requisitado e marca-se se a requisição foi atendida ou não, em ambos os casos em que o canal funciona, a cache muda de estado seguindo o padrão do tipo de cache especificado. Após a tentativa de se encaminhar a requisição para as duas caches, verifica-se se a requisição foi atendida por pelo menos uma das caches, se sim, é marcado como sucesso. Após uma requisição ser feita, é agendada uma nova requisição para o mesmo conteúdo onde o tempo do agendamento é gerado de forma aleatória seguindo a distribuição exponencial com parâmetro igual a probabilidade do conteúdo, dessa forma garantimos que cada conteúdo seja requisitado seguindo sua taxa de chegada.

Após o número de requisições pedidas serem feitas, verifica-se o número de casos em que não ocorreu user miss, ou seja, os eventos onde a requisição foi atendida e finalmente calcula-se a probabilidade de user miss fazendo o seguinte cálculo:

$$\frac{\textit{número de requisições realizadas} - \textit{número de casos de não user}}{\textit{número de requisições realizadas}}$$

- Parâmetros de entrada para a função do simulador

O simulador do cenário II é dado pela função “simulacaoCenario2()” que tem os seguintes parâmetros de entrada:

* numRequisicoes - Número de requisições que serão realizadas no experimento. Deve ser um número inteiro maior que 0.

* caso - qual é o caso/tipo das caches. Os possíveis valores de entrada são: "FIFO", "LRU", "Random" ou "Estatica"

* numConteudos - número de conteúdos que podem ser requisitados. Deve ser um número inteiro maior que 0.

* tamCache - tamanho das caches. Deve ser um número inteiro maior que 0.

* probabilidades - probabilidade de cada conteúdo ser requisitado. Deve ser passado uma lista de tamanho igual ao número de conteúdos. Por exemplo, se são três conteúdos e possuem probabilidade de requisição uniforme, a lista passada deverá ser [1/3,1/3,1/3]

* p - probabilidade de um canal falhar. Deve ser um float entre 0 e 1 e tem como default o valor 0.9.

* depurar - Booleano para indicar se deseja imprimir a depuração do código ou não. O default é o valor "False", onde não se depura.

* cache1_inicial - Estado inicial de uma cache. Se nada for passado, ela irá ser inicializada aleatoriamente.

* cache2_inicial - Estado inicial da outra cache. Se nada for passado, ela irá ser inicializada aleatoriamente.

* cachesDiferentes - Booleano para indicar se deseja que as caches comecem em estados diferentes ou iguais. O default é o valor "True", onde começam com em estados diferentes.

## 4.2. Resultados por Simulação

Em relação às simulações do cenário I, as probabilidades de sucesso, que no primeiro seria considerada falha, estão muito próximas.

Encontramos uma grande variação na probabilidade de sucessos no caso FIFO com 3 conteúdos, por depender do estado inicial. Já no caso com 4 conteúdos, novamente as probabilidades são estabilizadas.

No caso LRU, igualmente temos uma estabilização nos resultados. Conseguimos novamente comparar com o cenário I. Obtivemos, no atual, 34% de chance de sucesso, enquanto no cenário I, obtivemos 66% de chance. Esses valores de user miss e de uma requisição ser atendida,

respectivamente, se completam. Notamos que a chance de um canal falhar não influenciou muito nos resultados. O mesmo vale para o caso das caches estáticas que começam com estados iguais.

E o caso de caches Random se comportou analogamente, com a pequena diferença de que o sucesso no cenário II aumentou levemente. Isso ocorre porque ele, conforme visto no cenário I, tinha a melhor probabilidade de atender uma requisição, e não corria o risco de cair em estados iguais e nunca mais poder ficar novamente com estados diferentes. Dessa forma, a possibilidade de canais falharem atrapalha um pouco mais o atendimento de requisições. O mesmo vale para o caso das caches estáticas que começam com estados diferentes.

# 5. Cenário III – Com perdas e com atrasos

O cenário III também é composto por duas caches, cada uma com tamanho igual à dois, contudo apresenta atrasos nos canais de comunicação e também há a probabilidade de um canal falhar. Dessa forma, a cada requisição feita pelo cliente, primeiramente é avaliado se o canal funcionou e só assim é avaliado se a cache possui ou não o conteúdo requisitado e tem o tempo para a requisição chegar nas caches e nos servidores. Nosso propósito nesse cenário é identificar qual o tempo médio para uma requisição ser atendida nas seguintes combinações:

2 caches FIFO; 2 caches LRU; 2 caches Random; 2 caches Estáticas

## 5.1. Simulador

Para simularmos o cenário III, criamos um simulador de eventos de tempo contínuo muito mais complexo que os cenários anteriores, onde temos vários eventos para simular todo o processo que uma requisição passa até ser atendida, seja por uma das duas caches ou pelo servidor alternativo. Abaixo, temos uma melhor descrição do funcionamento do simulador e da função que o implementa.

### 5.1.1. Descrição do Simulador

O cenário III apresenta um funcionamento mais próximo da realidade. Nossa modelagem do problema funciona como descrito abaixo:

Temos que inicialmente, as caches são inicializadas com os estados passados, caso não sejam passados, elas são inicializadas de forma aleatória com os conteúdos possíveis, que nesse caso serão representados por números inteiros de 1 ao numConteudos dado.

Após isso, agenda-se um evento de uma requisição para cada conteúdo na lista de eventos, esse evento indica qual o tempo em que a requisição irá acontecer e qual conteúdo estará sendo requisitado. O tempo do agendamento é gerado de forma aleatória seguindo a distribuição exponencial tendo como parâmetro a taxa do conteúdo e a cada vez que um evento é adicionado, a lista é ordenada em função dos tempos apresentados em cada evento.

Com as caches inicializadas e uma requisição agendada para cada conteúdo, inicia-se o processo. Em termos de código, inicia-se um loop em cima da lista de eventos, onde para cada

requisição, novos eventos são gerados para simular seu atendimento. Primeiramente, tenta-se encaminhar a requisição com menor tempo presente na lista inicial de eventos para as duas caches. Aqui temos um novo evento e mais dois possíveis novos eventos para acrescentar na lista, pois tenta-se encaminhar a requisição para cada cache, para isso gera-se um número entre 0 e 1, se o número for menor ou igual ao parâmetro p (que por default é 0.9, o que implica que o canal tem 90% de chance de funcionar), o canal funciona e cria-se um novo evento da requisição chegando na cache, repete-se isso para as duas. Assim, há a possibilidade da criação de um evento onde a requisição chega na cache 1 e/ou outro evento onde a requisição chega na cache 2. Além disso, é gerado um número aleatório seguindo a distribuição exponencial com parâmetro $\frac{1}{\alpha}$ a para indicar o timer da requisição, assim, cria-se um novo evento que será o timeout da requisição.

Quando o evento da requisição chegando na cache acontecer, é verificado se a cache possui o conteúdo requisitado. Se sim, cria-se um novo evento da requisição sendo atendida pela cache agendado para acontecer em n segundos, onde n é calculado gerando um número aleatório seguindo a distribuição exponencial com parâmetro $\frac{1}{\mu}$. Já se a cache não possuir, primeiro cria-se um novo evento onde cache receberá o conteúdo requisitado do servidor para assim poder atender a requisição, o evento é agendado para acontecer em m segundos, onde m é calculado gerando um número aleatório seguindo a distribuição exponencial com parâmetro $\frac{1}{\theta}$ ($\theta = \infty$ no caso desse cenário) e só quando esse novo evento criado acontecer, que cria-se o evento da requisição sendo atendida.

Já se o evento timeout da requisição chegar a acontecer, da forma que modelamos, ele cancela todos os outros eventos referentes a requisição que sofreu o timeout, isso para simular o cancelamento do processo de tentativa de atender a requisição pela caches, e cria um evento da requisição sendo atendida pelo servidor alternativo, o evento é agendado para acontecer em l segundos, onde l é calculado gerando um número aleatório seguindo a distribuição exponencial com parâmetro $\frac{1}{\gamma}$, onde nesse cenário.

Por último, quando o evento requisição atendida acontece, seja pela cache 1, cache 2 ou pelo servidor alternativo por conta do timeout, verifica-se se há mais requisições abertas para o mesmo conteúdo da que está sendo atendida, e cancela todos os processos referentes a essas requisições junto com os processos referentes à que está sendo atendida, com exceção dos eventos intitulados “Sistema”, o que representa quando uma requisição será feita por um usuário e garante a taxa de chegada de cada tipo de conteúdo. Após isso, anota-se qual foi o tempo de atendimento de cada requisição fazendo tempo do evento requisição atendida - tempo de chegada da requisição.

Todos esses eventos acontecem para cada requisição e se misturam, ou seja, pode ter um evento da requisição "req2" chegando na cache 1 enquanto e logo em seguida outra requisição "req1" sendo atendida pelo servidor alternativo.

Após termos o número especificado pelo parâmetro de requisições atendidas, seja pelo jeito que for, as iterações param e calcula-se o tempo de atendimento médio das requisições fazendo-se a soma do tempo de atendimento de cada requisição/número de requisições.

Lembrando que sempre que uma nova requisição chega, outra requisição para o mesmo conteúdo é agendada onde o tempo do agendamento é gerado de forma aleatória seguindo a distribuição exponencial com parâmetro igual a probabilidade do conteúdo, dessa forma garantimos que cada conteúdo seja requisitado seguindo sua taxa de chegada. A depuração apresentada mais abaixo deve ajudar a entender o funcionamento da simulação aqui descrito que imaginamos que possa ser confusa já que muitos eventos são gerados. Após o número de requisições pedidas serem feitas,

verifica-se o número de casos em que não ocorreu user miss, ou seja, os eventos onde a requisição foi atendida e finalmente calcula-se a probabilidade de user miss fazendo o seguinte cálculo:

$$\frac{\textit{número de requisições realizadas} - \textit{número de casos de não user}}{\textit{número de requisições realizadas}}$$

- Parâmetros de entrada para a função do simulador

O simulador do cenário III é dado pela função abaixo que tem os seguintes parâmetros de entrada:

* numRequisicoes - Número de requisões que serão realizadas no experimento. Deve ser um número inteiro maior que 0.

* caso - qual é o caso/tipo das caches. Os possíveis valores de entrada são: "FIFO", "LRU", "Random" ou "Estatica"

* numConteudos - número de conteúdos que podem ser requisitados. Deve ser um número inteiro maior que 0.

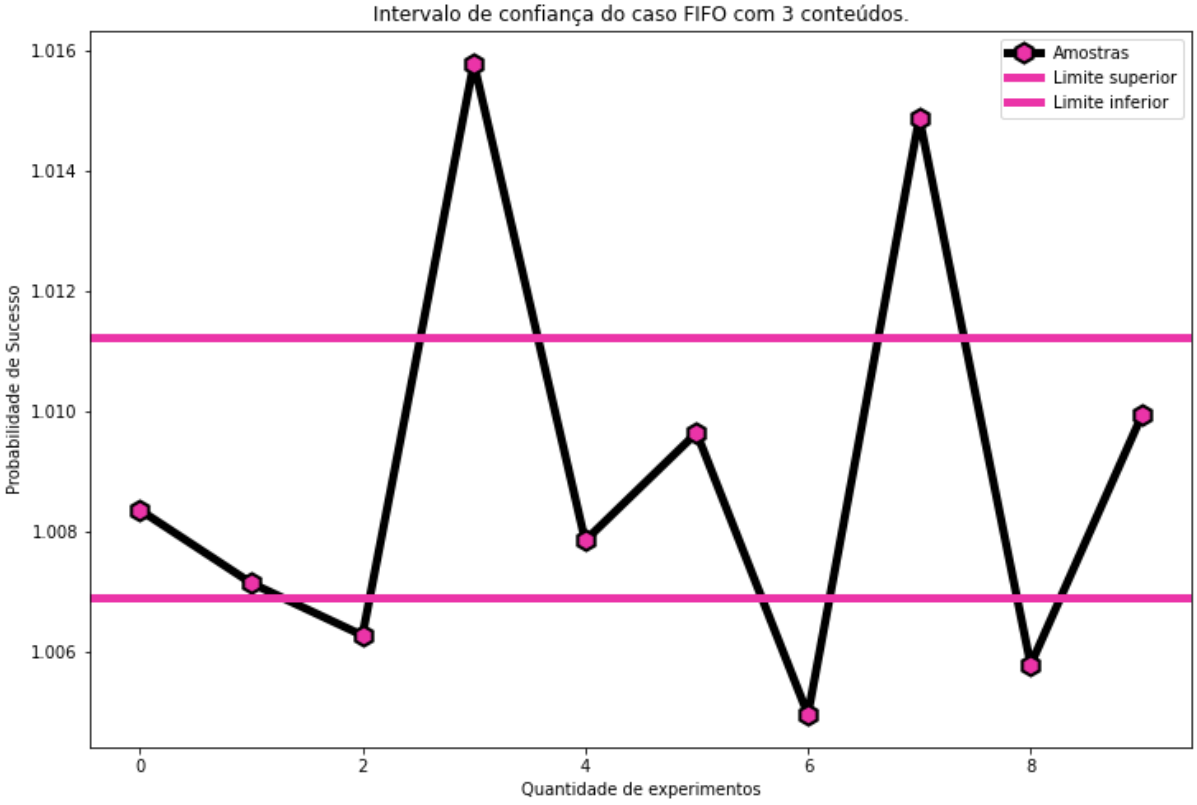
* tamCache - tamanho das caches. Deve ser um número inteiro maior que 0.

## 5.2. Resultados por Simulação

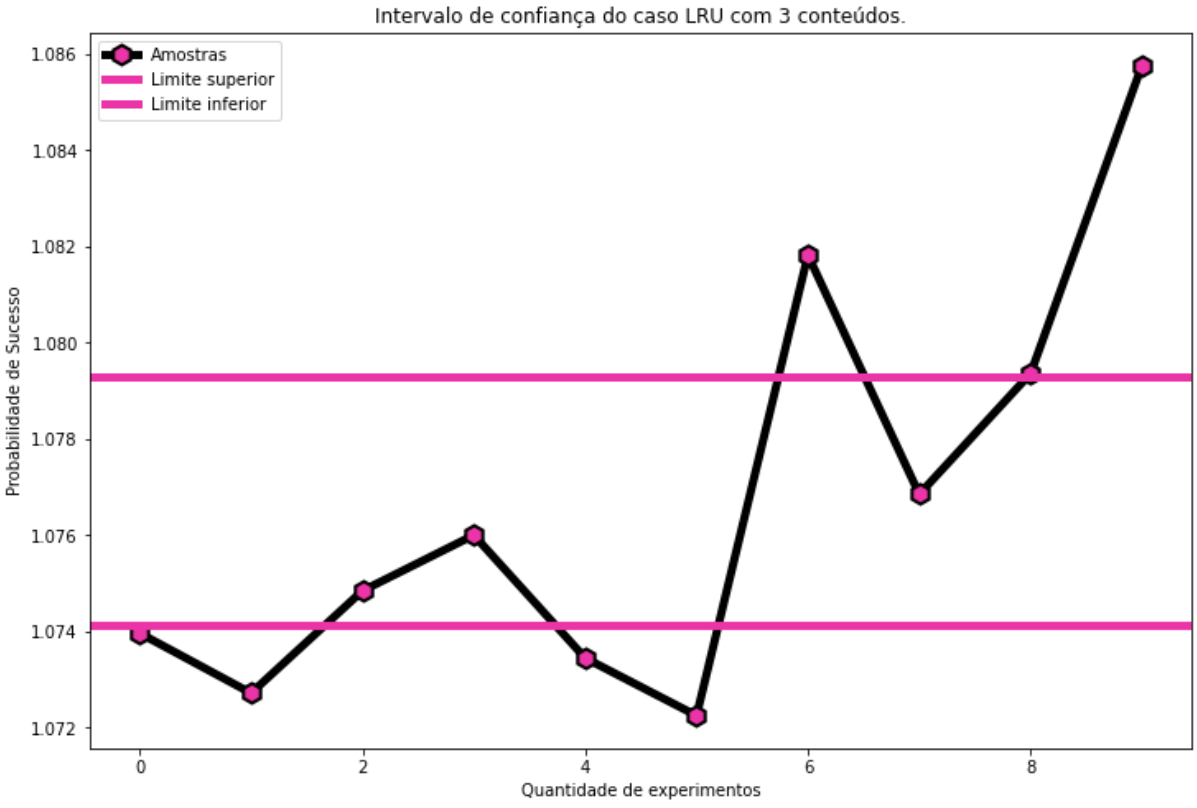
Ao executar nossa simulação com 10000 requisições, não encontramos diferenças significativas em relação ao tipo de cache utilizado. Todas elas gastaram em média 1 segundo, o que analisamos utilizando nossa função de intervalo de confiança.

Podemos observar esses resultados nos gráficos abaixo.

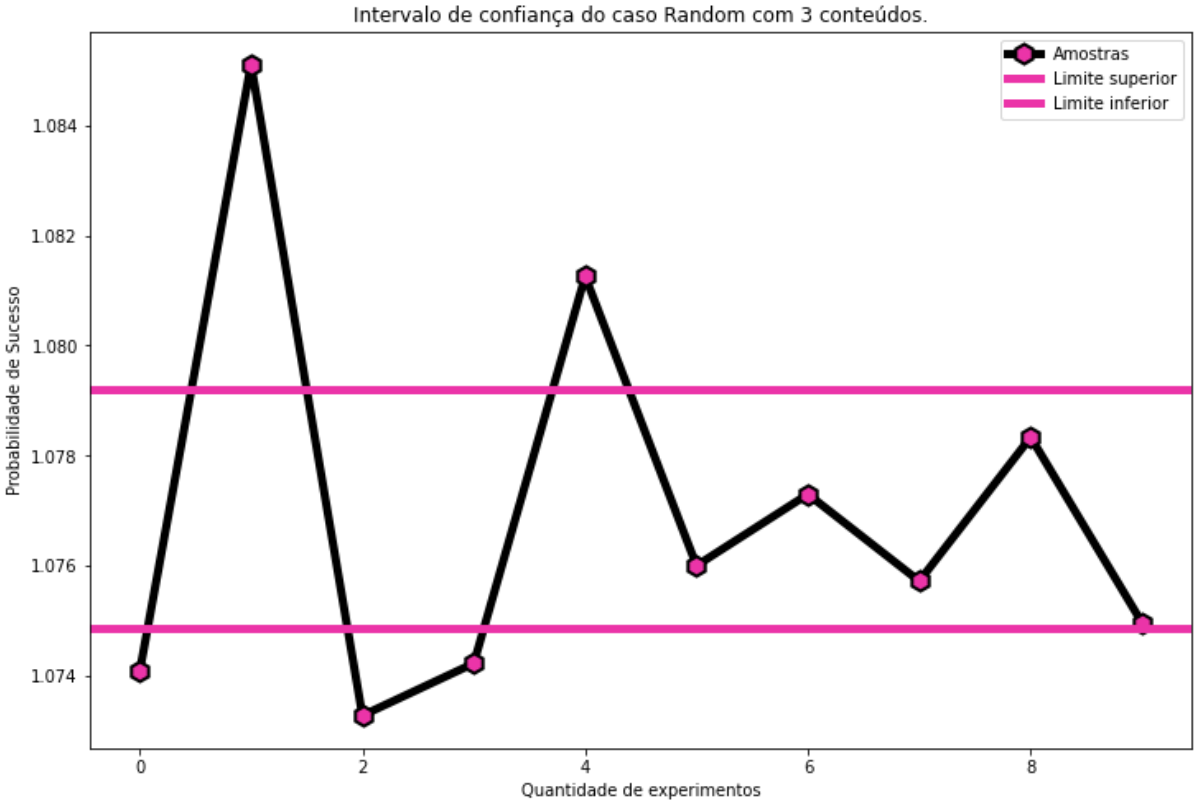
- FIFO


Intervalo de confiança do caso FIFO com 3 conteúdos.
Amostras
Limite superior
Limite inferior
Probabilidade de Sucesso
1.016
1.014
1.012
1.010
1.008
1.006
0
2
4
6
8
Quantidade de experimentos

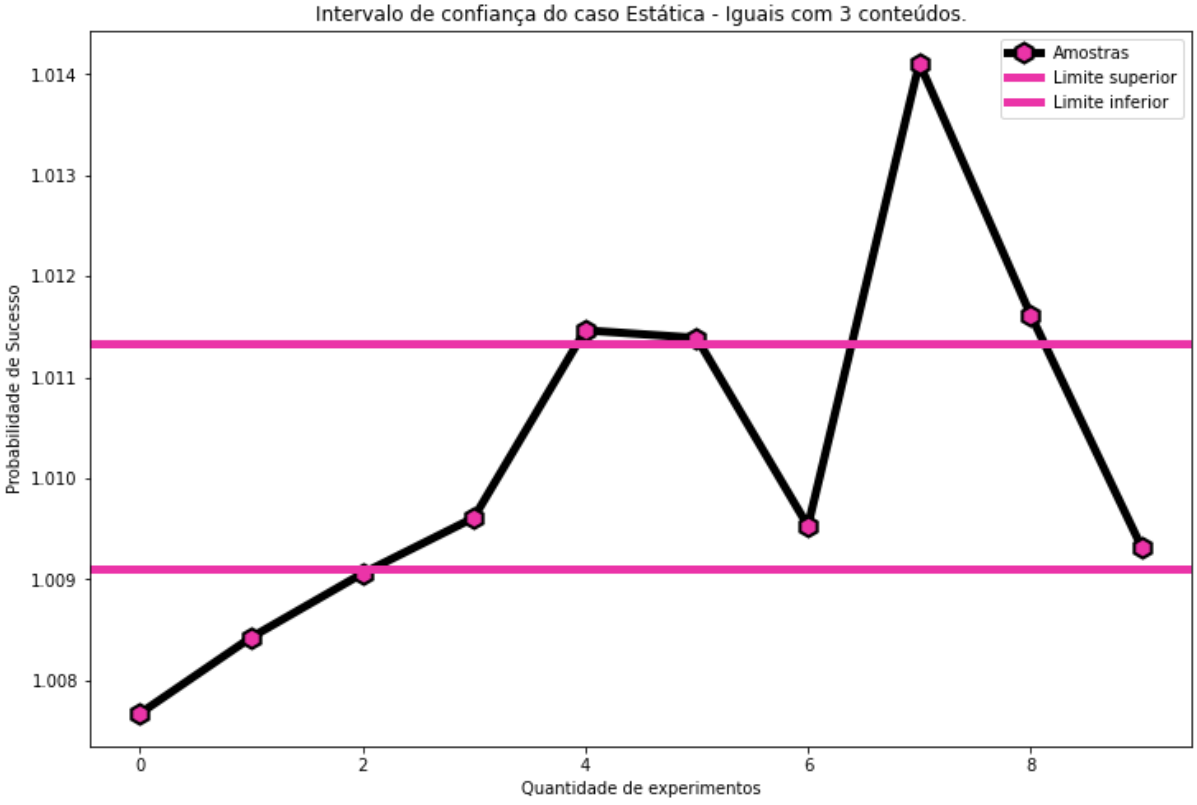
➢ LRU


Intervalo de confiança do caso LRU com 3 conteúdos.
Amostras
Limite superior
Limite inferior
Probabilidade de Sucesso
1.086
1.084
1.082
1.080
1.078
1.076
1.074
1.072
0
2
4
6
8
Quantidade de experimentos

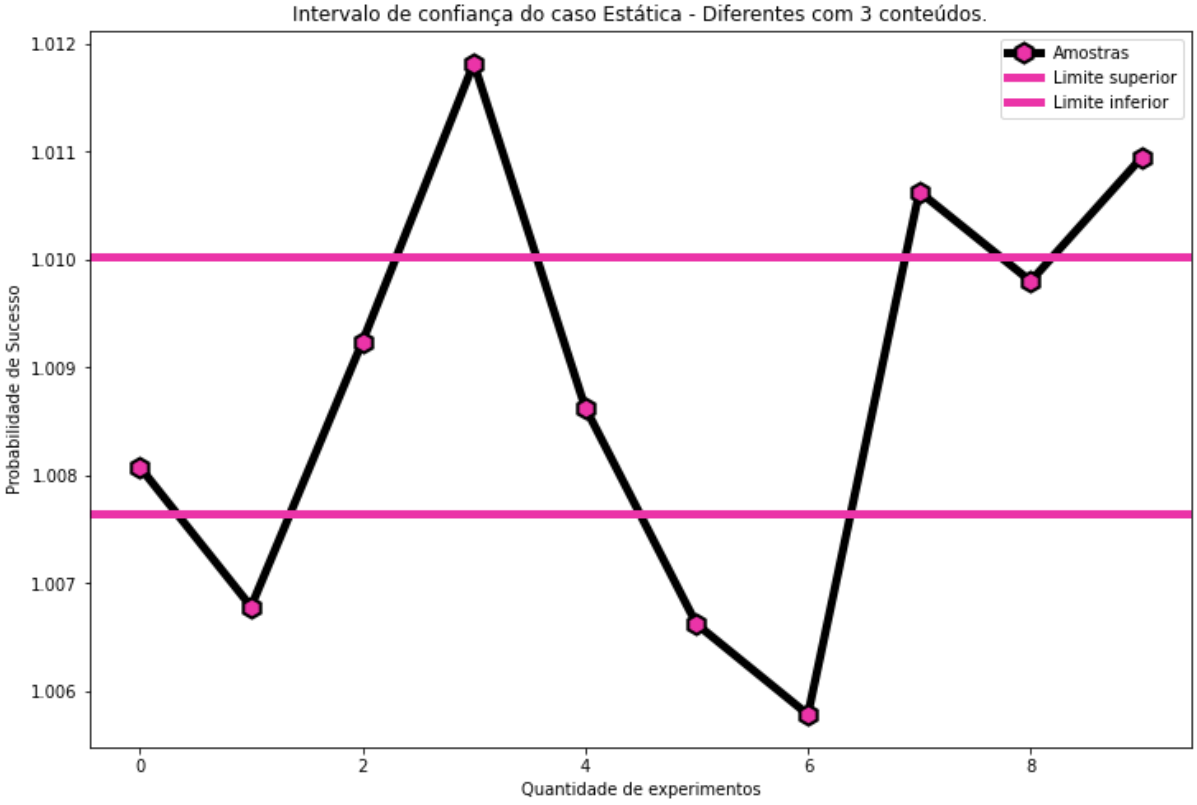
➢ Random




➢ Estática - começando iguais



➢ Estática - começando diferentes




# 6. Cenário IV – Com perdas e com atrasos

O cenário IV também é composto por duas caches, cada uma com tamanho igual à dois, contudo apresenta atrasos nos canais de comunicação e também há a probabilidade de um canal falhar. Extremamente parecido com o cenário III. Dessa forma, a cada requisição feita pelo cliente, primeiramente é avaliado se o canal funcionou e só assim é avaliado se a cache possui ou não o conteúdo requisitado e tem o tempo para a requisição chegar nas caches e nos servidores. Nosso propósito nesse cenário é identificar qual o tempo médio para uma requisição ser atendida nas seguintes combinações:

2 caches FIFO; 2 caches LRU; 2 caches Random; 2 caches Estáticas

## 6.1. Simulador

Para simularmos o cenário IV, utilizamos o mesmo simulador de eventos de tempo contínuo criado para o cenário III descrito acima e mudamos apenas o parâmetro θ, dessa forma, novamente temos vários eventos para simular todo o processo que uma requisição passa até ser atendida, seja por

uma das duas caches ou pelo servidor alternativo. A diferença do parâmetro implica numa taxa de comunicação entre uma cache e o servidor diferente de 0, então a cache não mais obtém o conteúdo do servidor de forma instantânea como acontece no cenário III.

## 6.2. Resultados para 3 conteúdos por Simulação

Ao refazermos a simulação do cenário 3 mudando apenas o parâmetro θ, pudemos notar, analisando cada tipo de cache com o intervalo de confiança, que não houve mudanças muito significativas tanto nas médias obtidas, quanto nos próprios intervalos.

Com isso, surgiu a interessante indagação: quais são os parâmetros que realmente influenciam no nosso cálculo das médias. Assim, surgiram as nossas ideias para o cenário adicional.

# 7. Cenário V – Adicional

Neste cenário, dividimos os nossos experimentos em 3. São eles: a variação do parâmetro α, a variação do parâmetro p, e a correlação de cada parâmetro. Em todos eles, também utilizamos pares de caches com tamanho 2, e 3 conteúdos. Nosso objetivo era encontrar quais os parâmetros seriam mais influentes.

## 7.1. Variação de α

Neste caso, utilizamos o simulador do cenário III. Experimentamos tornar α o mais alto possível, igualando a infinito, e também o mais baixo possível, igualando a 0.0000001, analisando a influência que o timeout exerce sobre nossa simulação. Com isso, percebemos que, com o maior α possível tende a diminuir o tempo médio nos experimentos. Entendemos com isso que o timeout se aproximaria de 0, e consequentemente o caminho que a requisição faria, seria ir direto para o servidor alternativo, gastando apenas um tempo exponencial com média 1 para ir e voltar. No caso FIFO, por exemplo, obtivemos uma média de 0,77 segundos com este valor do alfa, em contraposição com 1,24 segundos com o α baixo.

Concluímos, assim, que o valor ideal de α nesta modelagem do sistema seria então o máximo possível, em questão de desempenho.

## 7.2. Variação de p

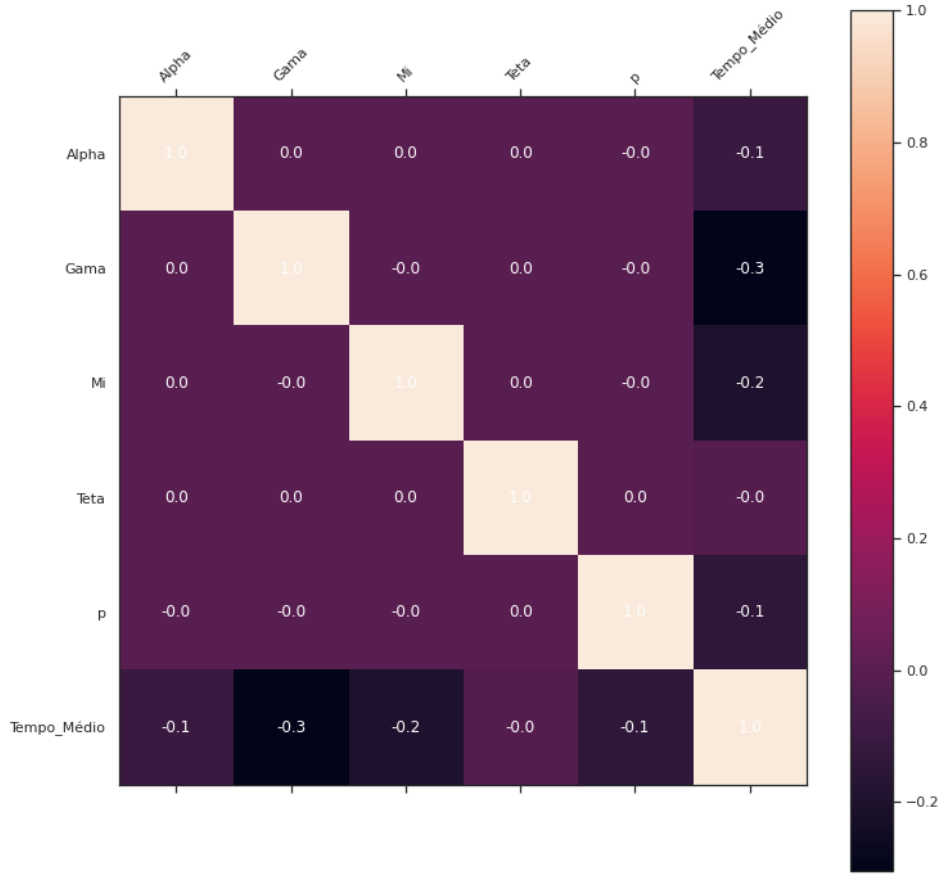
Utilizando a simulação do cenário 2, experimentamos variar a probabilidade de um canal funcionar entre os valores de 90%, 50% e 20%. Em todos os tipos de organização de cache, percebemos que esse valor não influenciou muito na média do tempo.

No caso FIFO, por exemplo, obtivemos uma média de 0,264 segundos com 90% de chance, 0,263 segundos com 50% e 0,258 segundos com 20% de chance.

Entendemos que essa pouca influência resultou do fato de a probabilidade de funcionamento do canal estar atrelado a uma única cache, e não necessariamente ambas as caches falharão juntas.

## 7.3. Correlação dos parâmetros

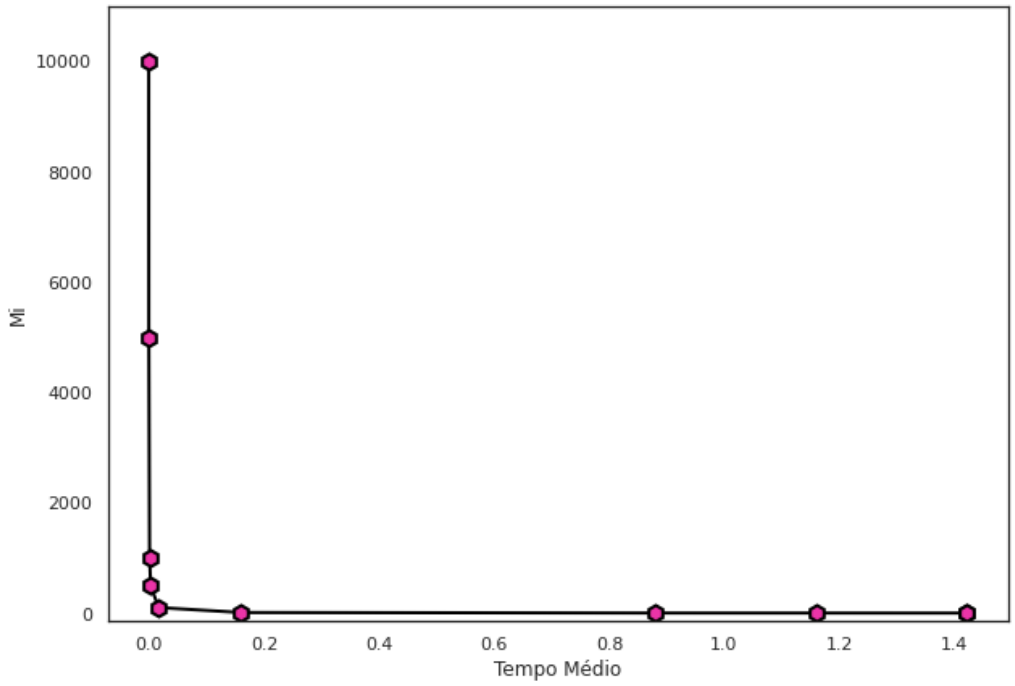
Simulamos várias vezes alterando todos os parâmetros entre 0.1, 1, 10, 100, 100000 para analisar seus comportamentos em relação ao tempo médio, e fizemos várias possíveis combinações, o que resultou em aproximadamente 1800 casos. Rodamos a simulação do cenário 3 com 10000 simulações, e assim construímos a matriz de correlação para descobrir quais parâmetros que mais afetam o tempo médio.



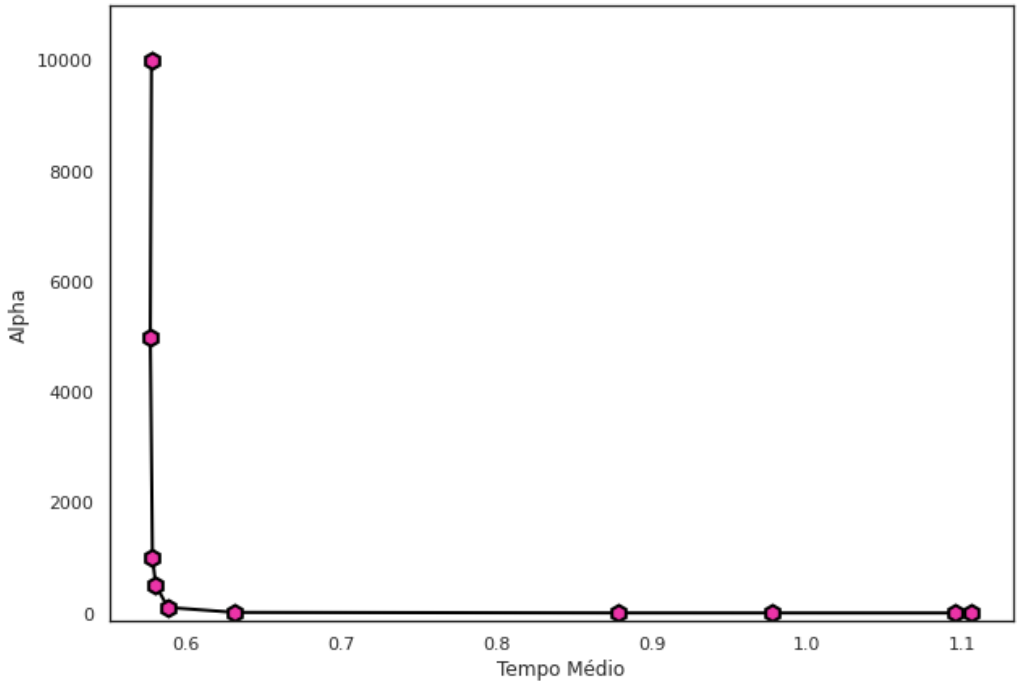

Pudemos ver que a mudança que ocorreu entre o cenário III e cenário III, influenciada pelo θ, novamente não foi muito significativa, o que mostrou que esse parâmetro não influenciou no tempo médio.

Podemos observar ambos esses resultados também com os gráficos separados:

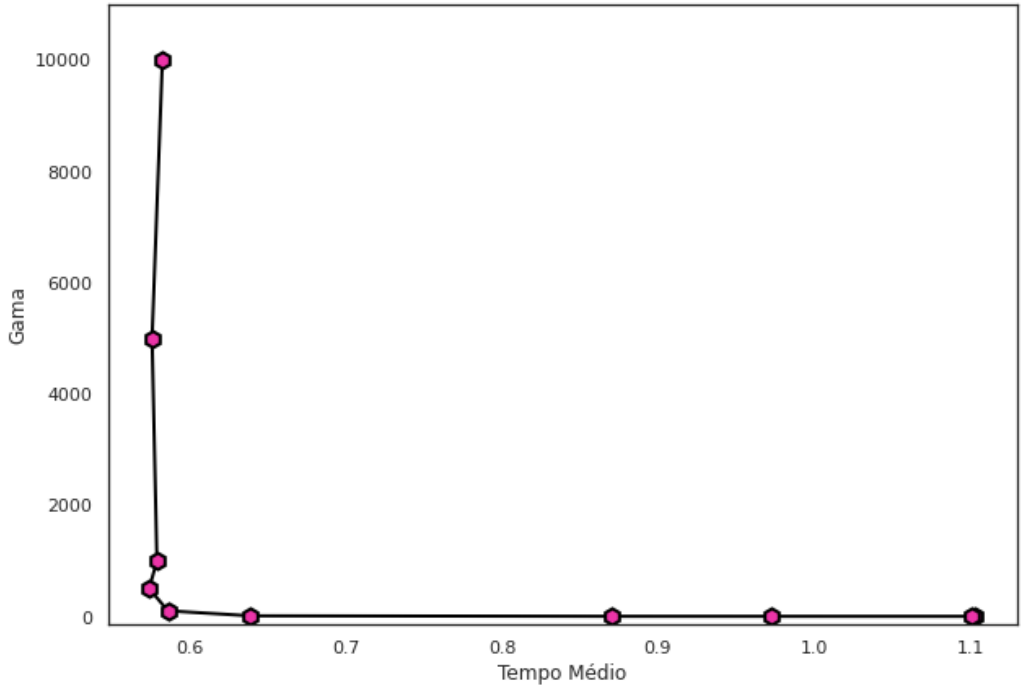
- μ


10000
8000
6000
4000
2000
0
Mi
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Tempo Médio

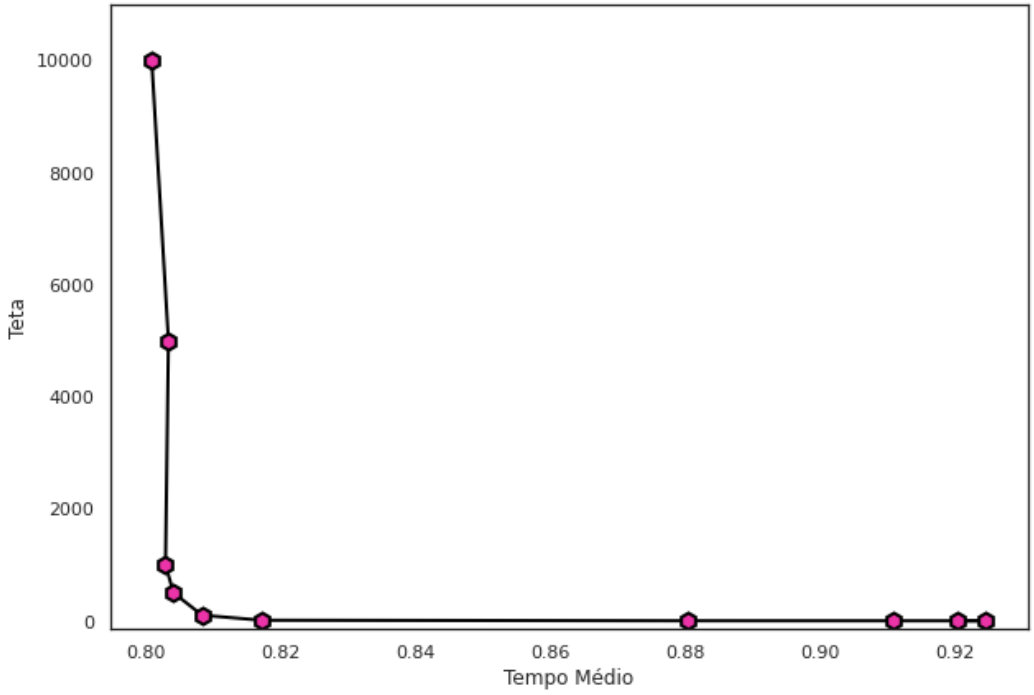
➢ α


10000
8000
6000
4000
2000
0
Alpha
0.6
0.7
0.8
0.9
1.0
1.1
Tempo Médio

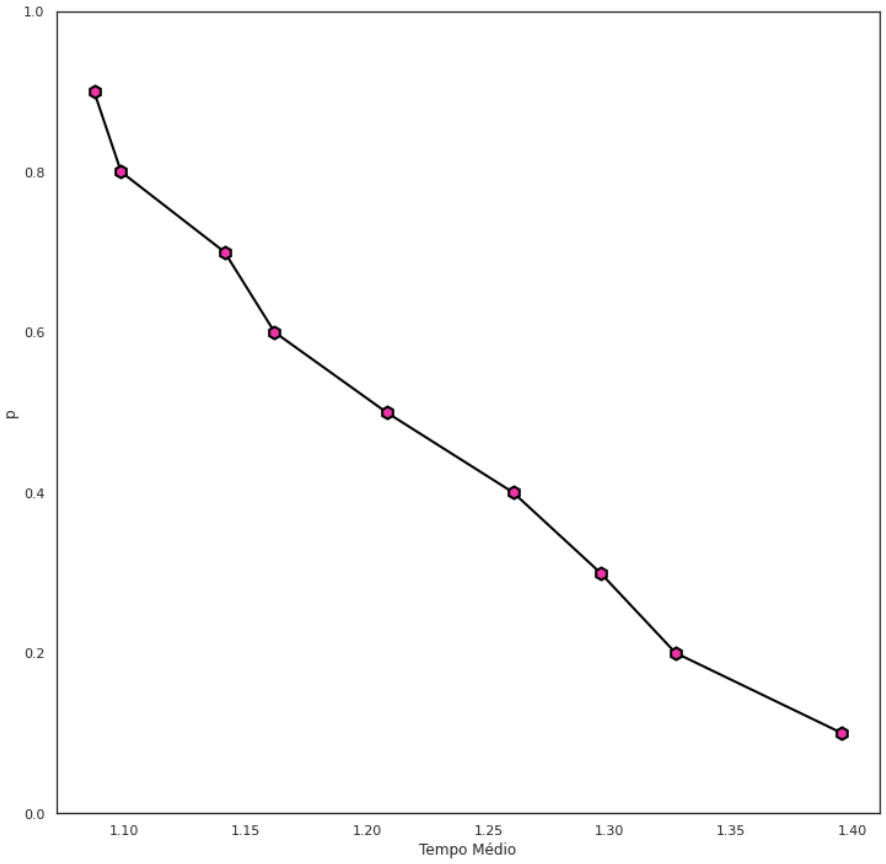
➢ γ


10000
8000
6000
4000
2000
0
Gama
0.6
0.7
0.8
0.9
1.0
1.1
Tempo Médio

➢ θ


10000
8000
6000
4000
2000
0
Teta
0.80
0.82
0.84
0.86
0.88
0.90
0.92
Tempo Médio

➢ p




Encontramos que os dois parâmetros que têm a maior correlação com o tempo médio são o $\gamma$, que representa o canal entre o usuário e o servidor alternativo, e o $\mu$, que representa o canal entre o usuário e a cache.

Pensamos então que, em casos reais, compensa-se muito mais utilizar caches em relação a acessar servidores diretamente. Então se desejássemos melhorar o tempo, preferiríamos alterar o $\mu$. A melhora do nosso sistema deveria vir dessa alteração da qualidade do canal entre o usuário e a cache.