Recomendação de Conteúdo e Desempenho de Sistemas de Cache

Raul G. M. de Freitas,[1] Daniel Menasché,[1] Carla Delgado[1] e Artur Ziviani[2]

[1] Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, RJ

[2]Laboratório Nacional de Computação Científica (LNCC), Petrópolis, RJ

Abstract. *Recommenders and cache systems affect millions of Internet users nowadays. On the one hand, an important share of the content demand from systems such as Netflix comes from recommendations generated by the Netflix platform itself. On the other hand, cache systems serve a significant portion of this content demand. Although content recommender and cache systems are two of the main components of the current Internet, the relation between them is not yet fully understood. In this paper, we propose models and optimization problems that aim at better understanding this relation. We show that some instances of the formulated problem have a simple closed-form solution. We also present preliminary results using Movielens traces. Our work is a step forward towards a better understanding of the mutual implications of recommender systems and caching algorithms, what we believe to be a promising front to further explore for benefiting both content providers and content consumers in the long run.*

Resumo. *Sistemas de recomendação e caches afetam milhões de usuários na Internet nos dias de hoje. Por um lado, uma demanda expressiva em sistemas como Netflix advém de recomendações feitas pela própria plataforma. Por outro lado, sistemas de cache servem uma significativa porção da demanda por conteúdo. Embora sistemas de recomendação e caches sejam dois dos componentes fundamentais da Internet atual, sua relação ainda não é bem entendida. Neste artigo, propomos modelos e problemas de otimização para melhor compreender essa relação. Mostramos que algumas instâncias do problema proposto admitem solução simples em fórmula fechada, enquanto que outras instâncias requerem solução numérica. Apresentamos também resultados preliminares usando* traces *do Movielens. Consideramos que nosso trabalho traz insumos para uma melhor compreensão das implicações mútuas de sistemas de recomendação e dos algoritmos de caching, o que acreditamos ser uma frente promissora a ser explorada no sentido de beneficiar provedores e consumidores de conteúdo no longo prazo.*

1. Introdução

Sistemas de recomendação de conteúdo e redes de cache são dois dos pilares da Internet do futuro. Estima-se que sistemas de recomendação sejam responsáveis por 80% das demandas no Netflix [Gomez-Uribe e Hunt, 2016], indicando que a essa demanda possui alta elasticidade [Turnbull, 2017]. Enquanto os sistemas recomendação moldam a demanda dos usuários por conteúdos, redes de cache afetam o desempenho da rede e consequentemente a qualidade de serviço (QoS) experimentada pelos usuários.

Neste artigo, apresentamos modelos e algoritmos que capturam o impacto dos sistemas de recomendação nos sistemas de cache. Para tal, estendemos o *framework* de maximização de utilidade do sistema recentemente proposto por [Dehghan et al., 2016]. Em tal *framework*, originalmente assume-se que as demandas dos usuários por conteúdos são dadas e fixas. Neste artigo, em contrapartida, assumimos que as demandas por conteúdos são as variáveis de controle. Fazemos então as seguintes perguntas:

1. Assumindo demandas elásticas por conteúdo, qual é o melhor sistema de recomendação de conteúdos para usuários, levando em conta o mecanismo de funcionamento das caches?
2. Qual o impacto da perturbação da demanda nas métricas de desempenho das caches?

Para responder a tais perguntas, formulamos problemas de otimização simples. Em seguida, adaptamos os problemas para determinados sistemas de cache que recentemente receberam atenção significativa na literatura, a saber TTL caches [Dehghan et al., 2016] e *reinforced counters* (RC) [Domingues et al., 2017]. Mostramos então que algumas das variantes dos problemas propostos admitem solução em fórmula fechada. Em resumo, nossas contribuições são:

1-Formulação do problema de recomendação ótimo: Formulamos um problema de otimização que visa ajustar parâmetros do sistema de recomendação, levando em conta características do sistema de cache. O problema é uma extensão daquele originalmente proposto em [Dehghan et al., 2016], sendo que agora as variáveis de controle passam a ser as taxas de requisição a cada conteúdo, controladas pelo sistema de recomendação.

2-Comparação de diferentes mecanismos de *caching*: Especializamos o problema de otimização para diferentes políticas de caching. Em particular, focando em caches do tipo TTL ou RC, o problema de otimização é consideravelmente simplificado. Em alguns casos, mostramos que o problema admite solução em fórmula fechada.

3-Avaliação de demanda inspirada por sistema real: Avaliamos o impacto do sistema de recomendação no desempenho do sistema, usando traces do Movielens [Miller et al., 2003].

O restante deste artigo está organizado da seguinte forma. A seção 2 descreve os conceitos fundamentais sobre os quais o trabalho foi construído. A seção 3 apresenta os modelos que utilizamos para calcular a probabilidade de hits em caches do tipo TTL e RC. Na seção 4 é apresentada nossa formulação do problema de otimização relacionando recomendação e desempenho, e na seção 5, a solução proposta para este problema. Os resultados obtidos usando dados reais para parametrização são mostrados na seção 6. A seção 7 faz uma breve revisão dos trabalhos relacionados, comparando-os com a nossa proposta, e a na seção 8 apresentamos nossas conclusões.

2. Fundamentos

2.1. Caches TTL e RC

Nesta seção, descrevemos brevemente o comportamento de caches *time to live* (caches TTL) e de caches baseados em *reinforced counters* (RC) [Dehghan et al., 2016, Domingues et al., 2017].

Caches TTL vêm recebendo significativa atenção na literatura, devido à sua simplicidade e à sua capacidade de replicar o comportamento de caches padrão, do tipo LRU e FIFO. Uma cache TTL associa, a cada conteúdo i, um contador T_i. O contador é decrementado a cada unidade de tempo. Quando o contador de um determinado conteúdo atinge o valor zero, esse conteúdo é removido da cache. Quando o conteúdo é requisitado novamente, esse conteúdo é re-inserido e o contador é então reiniciado ao valor T_i. Esse comportamento corresponde ao TTL sem reset.[1] O TTL com reset é similar. Entretanto, neste último caso toda vez que o conteúdo é requisitado, o contador é novamente setado para T_i, independente de o conteúdo estar ou não presente na cache (vide Figura 1).

Por sua vez, os caches baseados em *reinforced counters* (RCs) são variantes dos TTL caches [Domingues et al., 2015]. A cada vez que o conteúdo é acessado, o contador é incrementado em um. A cada intervalo de tempo ditado pelo temporizador, o contador é decrementado. Quando o contador alcança o valor zero, o conteúdo é removido da cache. Quando o conteúdo é requisitado novamente, o conteúdo é re-inserido e o contador é setado igual a um. Em caches TTL, o temporizador é decrementado a unidades de tempo fixo, iguais a uma unidade de tempo. Já em RCs, o temporizador do conteúdo i é decrementado em intervalos de tempo exponencialmente distribuídos, com taxa μ_i. Caches do tipo TTL e RC têm recebido atenção na literatura por três motivos fundamentais:

Simplicidade: Em caches do tipo TTL e RC, a dinâmica de inserção e remoção de cada conteúdo é completamente desacoplada dos outros conteúdos. Ou seja, para determinar quando um conteúdo será removido ou inserido novamente, basta conhecer-se os parâmetros daquele conteúdo. Em caches do tipo LRU e FIFO, por exemplo, precisa-se conhecer o estado completo da cache para determinar-se quando um conteúdo será removido ou re-inserido.

Aproximação de outras políticas: Usando caches TTL, pode-se aproximar outras políticas, como LRU e FIFO [Fofack et al., 2014, Dehghan et al., 2016]. Este resultado foi primeiramente observado em [Fagin, 1977] e redescoberto em [Che et al., 2009]. Desta forma, caso se deseje que uma cache se comporte como uma LRU ou FIFO, pode-se ainda assim adotar caches TTL, desde que seus parâmetros sejam ajustados adequadamente.

Adoção em sistemas reais: Caches TTL são utilizadas em sistemas do tipo DNS e ARP. Nesses sistemas, entradas expiram de tempos em tempos para permitir renovação, dado um prazo de validade. Caches TTL facilmente permitem que prazos de expiração sejam implementados.

2.2. Maximização de Utilidade

A cada conteúdo associamos uma função de utilidade. A função de utilidade pode levar em consideração os interesses dos usuários bem como o desempenho da cache. No caso mais geral, a utilidade associada a um conteúdo i depende da popularidade (taxa de requisição efetiva) do conteúdo, λ_i, da popularidade alvo (popularidade referência) estabelecida pelo sistema de recomendação de conteúdo, que leva em conta a natureza do conteúdo e os gostos dos usuários, $\tilde{\lambda}_i$, e da taxa de *hits* (*hit rate*), h_i. A função de utilidade é dada por $U_i(h_i, \lambda_i, \tilde{\lambda}_i)$.

[1]Note que numa cache TTL sem reset, o contador só é reiniciado quando o conteúdo é reinserido na cache.

Tabela 1. Notação adotada neste artigo.

Variável	Descrição
N	número de conteúdos
h_i	probabilidade de *hit* do conteúdo i
Λ_i	taxa de chegada de requisições ao conteúdo i, sem o sistema de recomendação
$\tilde{\lambda}_i$	taxa de chegada de requisições ao conteúdo i, perturbada pelo sistema de recomendação (não modificado)
λ_i	taxa de chegada de requisições ao conteúdo i, perturbada pelo sistema de recomendação proposto neste trabalho
T	tempo característico da cache (para LRU e FIFO)
B	tamanho da cache
ϵ	limiar de aceitação do nível de perturbação do sistema de recomendação modificado
$U_i(\lambda_i, \tilde{\lambda}_i, h_i)$	função de utilidade associada ao conteúdo i
p_{ij}	variável de controle probabilidade de recomendar j para um usuário que requisita i
c_{ij}	probabilidade do sistema de recomendação (puro) recomendar j para um usuário que requisita i

Em [Dehghan et al., 2016] assume-se que T_i, o parâmetro básico da cache TTL, é a variável de controle. Já em [Domingues et al., 2015], assume-se que pode-se controlar a taxa de decréscimo do temporizador, μ_i. Tanto T_i quanto μ_i afetam h_i, que por sua vez afeta a função de utilidade. Neste trabalho, em contrapartida, assume-se que T_i ou μ_i são dados, e que a variável de controle, influenciada pelo sistema de recomendação, é a taxa de chegada de requisições ao conteúdo, λ_i.

3. Modelos de Caches TTL e RC

Nesta seção, introduzimos modelos para determinar a probabilidade de *hit* em caches TTL e RC.

3.1. Quantidades Básicas

Consideramos um conjunto de N conteúdos. Seja Λ_i a taxa de chegada de requisições ao conteúdo $i = 1, \ldots, N$, sem levar em consideração o sistema de recomendação. Para cada conteúdo i, seja h_i a probabilidade de *hit* correspondente.

Seja H_i a variável aleatória que determina se o conteúdo i está na cache, ou seja,

$$H_i = \begin{cases} 1, & \text{conteúdo } i \text{ na cache,} \\ 0, & \text{caso contrário.} \end{cases} \tag{1}$$

Então, h_i é dado pelo valor esperado de H_i,

$$h_i = \mathbb{P}(H_i = 1) = \mathbb{E}[H_i], \quad i = 1, \ldots, N. \tag{2}$$

A Tabela 1 resume a notação considerada neste artigo.

3.2. Restrição na Ocupação Média

Assumimos que todos os conteúdos de interesse tem o mesmo tamanho (esta hipótese pode ser facilmente relaxada). Com isso, consideramos que a cache possui uma capacidade um pouco maior que B, suficiente para armazenar B conteúdos e outros mais. A fim

de simplificar a análise, assumimos que a restrição sobre a ocupação da cache é dada em termos do número médio de conteúdos armazenados. Ou seja, assumimos que em alguns momentos a cache pode conter mais que B elementos, e em outros momentos menos que B, mas na média a cache deve conter B elementos. Desta forma, assumimos que

$$\sum_{i=1}^{N} h_i = B. \tag{3}$$

Note que h_i é a fração de tempo que o conteúdo i passa na cache. Logo, $\sum_{i=1}^{N} h_i$ corresponde ao número médio B de conteúdos na cache.

Na prática, muitas caches atualmente já possuem a elasticidade para armazenar dinamicamente mais ou menos conteúdos ao longo do tempo. O Amazon ElastiCache é um exemplo deste tipo de mecanismo [Amazon, 2015]. Alternativamente, mostrou-se em [Dehghan et al., 2016] que dimensionando a cache para conter um tamanho um pouco maior que B, ou seja, $B(1+\tilde{\epsilon})$, pode-se garantir uma baixa probabilidade de *overflow*, logo incorrendo em um pequeno *overhead*.

3.3. Caches TTL

Consideremos primeiramente caches com reset. Uma vez que o conteúdo i é removido, ele será re-inserido após tempo médio $1/\lambda_i$. Uma vez inserido, com probabilidade $e^{-\lambda_i T_i}$ a próxima requisição ocorrerá depois do intervalo T_i e resultará em um *miss*. Caso contrário, resultará em um *hit* e um novo ciclo é iniciado. Logo,

$$h_i = 1 - e^{-\lambda_i T_i}. \tag{4}$$

Consideremos agora caches sem reset. Nesse caso, uma vez que o conteúdo é inserido, as próximas $\lambda_i T_i$ requisições, em média, irão gerar hits. Logo, a fração de hits é

$$h_i = 1 - \frac{1}{1+\lambda_i T_i} = \frac{\lambda_i T_i}{1+\lambda_i T_i}. \tag{5}$$

3.4. Caches RC

Caches RC são modeladas por cadeias de Markov do tipo nascimento e morte. No caso mais simples, assumimos que quando o contador atinge o valor zero, o conteúdo é removido da cache. Então, a probabilidade de hit é dada pela probabilidade de a cadeia de Markov estar em qualquer estado diferente de zero. Logo,

$$h_i = \lambda_i/\mu_i. \tag{6}$$

A Figura 1 ilustra o comportamento das caches TTL e RC.

3.5. Caches LRU e FIFO: Tempos Característicos

Caches do tipo LRU e FIFO funcionam como *filtros passa baixa*. Observa-se que o limiar de filtragem de tais caches é dado por uma constante T, chamada de tempo característico da cache. Conteúdos i, tais que $\lambda_i < 1/T$, tem baixa popularidade e são ignorados pela cache. Conteúdos tais que $\lambda_i > 1/T$, por outro lado, ficam na cache com grande

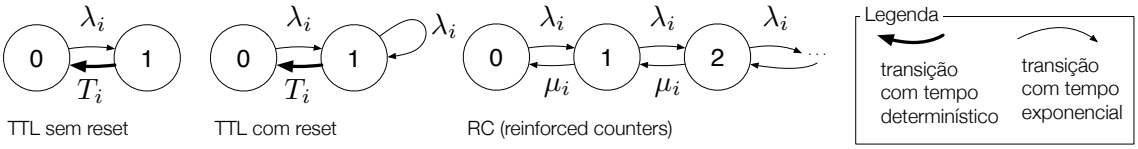

Figura 1. Caches TTL e RC.

probabilidade. O tempo característico T da cache é determinado pela solução do seguinte problema de ponto fixo:

$$\sum_{i=1}^{N} 1 - e^{-\lambda_i T} = B \tag{7}$$

para caches do tipo LRU, e

$$\sum_{i=1}^{N} \frac{\lambda_i T}{1 + \lambda_i T} = B \tag{8}$$

para caches FIFO.

Dado o vetor de demandas $(\lambda_1, \ldots, \lambda_N)$, pode-se determinar o tempo característico da cache, bem como as demais métricas de interesse. Logo, para caches LRU,

$$h_i = 1 - e^{-\lambda_i T} \tag{9}$$

e, para caches FIFO,

$$h_i = 1 - \frac{1}{1 + \lambda_i T} = \frac{\lambda_i T_i}{1 + \lambda_i T}. \tag{10}$$

3.6. Sistema Original e Perturbado

Os resultados apresentados nesta seção permitem-nos obter a probabilidade de hit para cada conteúdo, sabendo-se o tamanho da cache B e o vetor de demandas $(\lambda_1, \ldots, \lambda_N)$.

Assumimos que o vetor de demandas original, sem perturubação pelo sistema de recomendação, é dado por $(\Lambda_1, \ldots, \Lambda_N)$. O sistema de recomendação perturba tal vetor, dando origem ao vetor $(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_N)$.

Seja c_{ij} a probabilidade do sistema de recomendação indicar o conteúdo j para um usuário que requisita o conteúdo i. Neste trabalho, assumimos que

$$\tilde{\lambda}_i = \sum_{j=1}^{N} \Lambda_j c_{ji}. \tag{11}$$

Finalmente, seja p_{ij} a probabilidade do sistema de recomendação modificado, proposto neste trabalho, indicar o conteúdo j para um usuário que requisita i. Seja λ_i a demanda efetiva pelo conteúdo i, logo

$$\lambda_i = \sum_{j=1}^{N} \Lambda_j p_{ji}. \tag{12}$$

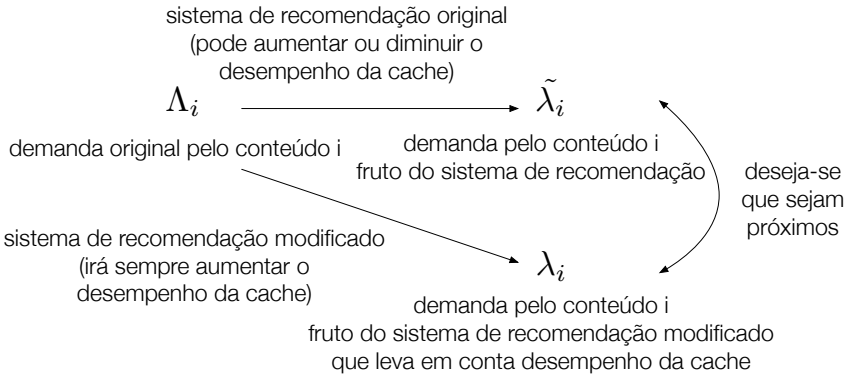

Figura 2. Perturbação da demanda pelo sistema de recomendação

Em princípio, desejamos que λ_i não difira em muito de $\tilde{\lambda}_i$ (vide Figura 2). Afinal, o sistema de recomendação original é calibrado de tal forma a satisfazer as demandas dos usuários, levando em conta seus gostos por conteúdos. Entretanto, damos uma certa flexibilidade para que o sistema leve em conta características de desempenho, como o funcionamento das caches, a fim de perturbar a demanda de forma estratégica. Na seção seguinte, definimos os problemas de otimização considerados no restante deste trabalho.

4. Formulação do Problema de Otimização

A seguir, estabelecemos o problema de otimização em sua forma mais geral. Consideramos que a cada conteúdo i está associada uma função de utilidade $U_i(h_i, \lambda_i, \tilde{\lambda}_i)$. A função de utilidade leva em conta o desempenho da rede, por meio da probabilidade de *hit*, h_i, bem como o desvio da demanda, λ_i, em relação ao valor padrão estabelecido pelo algoritmo de recomendação, $\tilde{\lambda}_i$.

$$\max \sum_{i=1}^{N} U_i(h_i, \lambda_i, \tilde{\lambda}_i) \tag{13}$$

$$\sum_{i=1}^{N} h_i = B \tag{14}$$

$$h_i = \varphi_i(\lambda_i) \tag{15}$$

$$\lambda_i = \sum_{j=1}^{N} \Lambda_j p_{ji} \tag{16}$$

$$\phi_{ij}(p_{ji}, c_{ji}) \leq \epsilon, \quad i = 1, \ldots, N, j = 1, \ldots, N \tag{17}$$

Note que o valor de p_{ii} pode ser setado de acordo com as hipóteses do problema. Por exemplo, pode-se assumir que $p_{ii} = 1$. Assim, capturamos o fato de que um usuário irá sempre assistir primeiro o conteúdo requisitado, e em seguida um conteúdo recomendado pelo sistema de recomendação. Nesta seção e nas próximas, não fazemos nenhuma restrição sobre o valor de p_{ii}. Em particular, um usuário que requisita o conteúdo i pode eventualmente nunca vir a assistir tal conteúdo. Para as outras entradas da matriz P,

assumimos que

$$\sum_{i=1,i\neq j}^{N} p_{ji} \leq 1 \tag{18}$$

A função objetivo (13) captura o fato de que se deseja maximizar a soma das funções utilidade associadas a cada conteúdo. A restrição (14) refere-se à ocupação esperada da cache, que não deve exceder B. A restrição (15) determina a probabilidade de *hit* em função do mecanismo de cache adotado. Esta equação deve ser substituída por alguma dentre as equações (4), (5) ou (6).

As restrições (16) e (17) são opcionais, e detalhes relacionados às probabilidades associadas ao sistema de recomendação. A equação (16) determina λ_i em função do sistema de recomendação. Note que, dependendo do nível de detalhe que se deseje capturar, esta equação pode ser ignorada, uma vez que determinados os λ_i pode-se buscar, separadamente, um conjunto de p_{ji} que resolvam (16). Finalmente, o conjunto de M equações (17) capturam restrições sobre a disparidade entre p_{ji} e c_{ji}. Por exemplo, para se capturar o fato de que p_{ji} e c_{ji} devem ser próximos, pode-se estabelecer

$$\phi_{ij}(p_{ji}, c_{ji}) = |p_{ji} - c_{ji}| \tag{19}$$

Alternativamente, pode-se substituir algumas das restrições estritas por funções de penalidade. Por exemplo, a restrição (14) pode ser substituída por um termo extra na função de utilidade, que passa a ser

$$\max \sum_{i=1}^{N} U_i(h_i, \lambda_i, \tilde{\lambda}_i) - \beta \left(\sum_{i=1}^{N} h_i - B \right) \tag{20}$$

Esta é uma estratégia padrão usada para resolver problemas de otimização convexos. Esta abordagem é inspirada no método Lagrangiano [Boyd e Vandenberghe, 2004], que iremos adotar na seção a seguir para resolver algumas instâncias simples do problema em questão.

5. Solução do Problema de Otimização

A seguir, resolvemos instâncias particulares do problema de otimização proposto na última seção. Em particular, assumimos que a função de utilidade captura o erro médio quadrático entre a demanda λ_i e a demanda referência $\tilde{\lambda}_i$,

$$U_i(\lambda_i, \tilde{\lambda}_i) = -\sum_{i=1}^{N} (\lambda_i - \tilde{\lambda}_i)^2 \tag{21}$$

Além disso, ignoramos as restrições (16) e (17).

Para simplificar a apresentação, assumimos que a variável de controle é λ_i, assumindo que p_{ij} pode ser ajustada, a posteriori, de acordo. Desta forma, o problema de otimização passa a ser

$$\min_{\boldsymbol{\lambda}} \sum_{i=1}^{N} (\lambda_i - \tilde{\lambda}_i)^2 \tag{22}$$

$$\sum_{i=1}^{N} \varphi_i(\lambda_i) = B \tag{23}$$

O Lagrangiano é dado por

$$\mathcal{L}(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}, \beta) = \sum_{i=1}^{N} (\lambda_i - \tilde{\lambda}_i)^2 - \beta \left(\sum_{i=1}^{N} \varphi_i(\lambda_i) - B \right) \tag{24}$$

5.1. Caches RC

Nesta seção, consideramos caches RC. Neste caso, aplicando (6) em (15), temos que

$$\varphi_i(\lambda_i) = \lambda_i / \mu_i \tag{25}$$

onde assumimos que os μ_i, $i = 1, \ldots, N$, são dados e fixos. Neste caso,

$$\mathcal{L}(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}, \beta) = \sum_{i=1}^{N} (\lambda_i - \tilde{\lambda}_i)^2 - \beta \left(\sum_{i=1}^{N} \lambda_i / \mu_i - B \right) \tag{26}$$

Derivando o Lagrangiano com relação a λ_i e igualando o resultado a zero, obtemos

$$\lambda_i = \frac{2\tilde{\lambda}_i - \beta/\mu_i}{2} \tag{27}$$

$$\beta = 2 \frac{\left(\sum_{i=1}^{N} \tilde{\lambda}_i / \mu_i \right) - B}{\sum_{i=1}^{N} 1/\mu_i^2} \tag{28}$$

Note que não escrevemos explicitamente as restrições de não-negatividade relativas a λ_i. Em muitos casos práticos, observamos que as variáveis de controle naturalmente assumem valores positivos. Caso contrário, pode-se facilmente ajustar a solução do problema para lidar-se com tais restrições [Boyd e Vandenberghe, 2004].

Exemplo numérico: Considere $N = 3$ conteúdos, cuja demanda é inspirada pela distribuição Zipf, $\tilde{\boldsymbol{\lambda}} = (1, 1/2, 1/3)$. Seja $B = 1$ (cache só suporta armazenar, em média, um conteúdo), e $\boldsymbol{\mu} = (1, 1, 1)$. Então, a solução do problema acima é dada por $\boldsymbol{\lambda} = (13/18, 4/18, 1/18)$. O conteúdo mais popular continua recebendo requisições a taxa próxima a 1, enquanto que os demais conteúdos tem suas taxas de requisições substancialmente diminuidas.

5.2. Caches TTL com reset

Nesta seção, consideramos caches TTL com reset. Neste caso, aplicando (4) em (15), temos que

$$\varphi_i(\lambda_i) = 1 - e^{-\lambda_i T_i} \tag{29}$$

onde assumimos que os T_i, $i = 1, \ldots, N$, são dados e fixos. Neste caso, o Lagrangiano é dado por

$$\mathcal{L}(\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}, \beta) = \sum_{i=1}^{N} (\lambda_i - \tilde{\lambda}_i)^2 - \beta \left(\sum_{i=1}^{N} 1 - e^{-\lambda_i T_i} - B \right) \tag{30}$$

Derivando o Lagrangiano com relação a λ_i e igualando o resultado a zero, obtemos

$$\lambda_i = 1 + \frac{W(T_i^2 \beta e^{-T_i \tilde{\lambda}_i})}{T_i} \tag{31}$$

$$\sum_{i=1}^{N} 1 - e^{-\lambda_i T_i} = B \tag{32}$$

onde $W(x)$ é a função W de Lambert, tal que $w = W(x)$ se $we^{-w} = x$. A função W de Lambert não admite expressão analítica. Dados os valores de $T_i, \tilde{\lambda}_i$, para $i = 1, \ldots, N$, e B, o valor de β que resolve o problema (31)-(32) nesse caso precisa ser obtida numericamente, por exemplo, usando um algoritmo de ponto fixo.

5.3. Caches TTL sem reset

Consideremos agora caches TTL sem reset. Neste caso, aplicando (5) em (15), temos que

$$\varphi_i(\lambda_i) = \frac{\lambda_i T_i}{1 + \lambda_i T_i} \tag{33}$$

Neste caso, pode-se escrever o Lagrangiano (24) de forma análoga a realizada nas últimas duas seções. Entretanto, ao derivarmos o Lagrangiano com relação a λ_i não fomos capazes de derivar uma expressão simples para λ_i, e acreditamos que o problema neste caso não admite solução em fórmula fechada.

6. Parametrizando Modelos com MovieLens

Nesta seção, ilustramos os benefícios do sistema de recomendação no desempenho de caches usando dados reais, advindos do *trace* MovieLens [Miller et al., 2003]. Os resultados preliminares aqui apresentados servem para ilustrar como que, em um sistema real, o sistema de recomendação pode beneficiar a taxa de *hits* (*hit rate*). Consideramos um *dataset* com 100,000 avaliações de 1,000 usuários a N=1,700 filmes.

O MovieLens é uma plataforma de recomendação de filmes, onde os usuários avaliam os filmes assistidos com notas (de 1 a 5 estrelas, na escala Likert). Usamos este *trace* para parametrizar a demanda original a cada filme i, que denotamos por Λ_i, e a demanda perturbada pelo sistema de recomendação, que denotamos por $\tilde{\lambda}_i$. Em seguida, comparamos o *hit rate* a uma cache alimentada pela demanda original e pela demanda perturbada, e verificamos que de fato a perturbação sugerida a seguir melhora o desempenho do sistema.

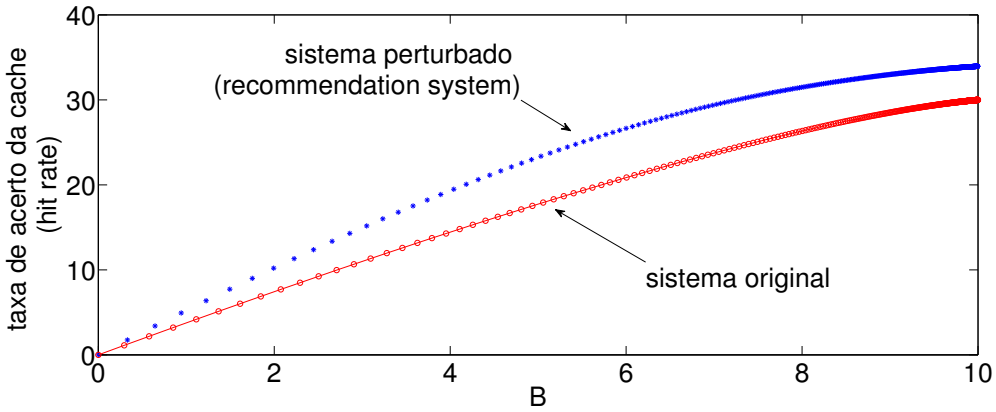

Figura 3. Desempenho da cache é aumentado com sistema de recomendação (curva azul) em comparação com sem recomendação (curva vermelha).

Usamos uma notação similar a aquela proposta em [Munaro et al., 2015]. Seja $n_{u,i}$ a nota que o usuário u deu ao filme i. Seja $n_{u,k,i}$ a nota que o usuário u deu ao filme i, quando o usuário previamente assistiu ao filme k. Seja m_k o número de notas dadas ao filme k, e seja $m_{k,i}$ o número de notas dadas ao filme i por usuários que previamente assistiram o filme k. Seja $I_{u,i}$ (resp., $I_{u,k,i}$) uma variável indicadoras, igual a 1 se $n_{u,i} \geq 3$ (resp., $n_{u,k,i} \geq 3$), e 0 caso contrário.

Recordamos que Λ_k é a popularidade do filme k. Assumindo que um usuário só avalia um filme que ele tenha assistido, parametrizamos Λ_k a partir de m_k da seguinte forma,

$$\Lambda_k = \frac{m_k}{\sum_{k=1}^{N} m_k} \tag{34}$$

Construímos então uma matriz C de correlação entre filmes, cujo elemento $c_{i,j}$ determina a probabilidade do sistema de recomendação indicar o filme j a um usuário que assistiu o filme i. Neste trabalho, assumimos que $c_{i,j}$ é dado por

$$c_{i,j} = \frac{\sum_{\forall u} I_{u,i,j}}{\sum_{\forall u} \sum_{l=1}^{N} I_{u,i,l}} \tag{35}$$

A equação acima captura o fato de que o sistema de recomendação procura recomendar o filme j para um usuário que assistiu i caso previamente usuários que tenham assistido i tenham avaliado bem o filme j.

Parametrizamos então a taxa de requisições ao conteúdo k, perturbada pelo sistema de recomendação, como

$$\tilde{\lambda}_k = \sum_{j=1}^{N} \Lambda_j c_{j,k} \tag{36}$$

Usamos os resultados apresentados na Seção 3.5 para calcular o *hit rate* associado às demandas originais e modificadas pelo sistema de recomendação. Variamos a ocupação média do buffer B, e apresentamos o *hit rate* da cache em função de B na Figura 3 (resultados numéricos adicionais encontram-se no relatório técnico, disponível em `https://tinyurl.com/recwperf17`). O uso do sistema de recomendação

aumenta significativamente o desempenho (*hit rate*) da cache. Isto deve-se ao fato de que, com o sistema de recomendação considerado, a curva de demanda torna-se mais enviesada, favorecendo os filmes mais populares e em consequência aumentando a disponibilidade dos conteúdos mais requisitados.

Nesta seção, assumimos que o sistema de recomendação leva em conta apenas a natureza dos conteúdos, e que não ele não é influenciado pelo desempenho das caches. Por isso, focamos nas taxas Λ_i e $\tilde{\lambda}_i$, e não na parametrização de λ_i (vide Tabela 1). É interessante observar que, mesmo sem levar em conta explicitamente o desempenho das caches, o sistema de recomendação já impacta positivamente o desempenho das mesmas. Com a recomendação estratégica de conteúdo, levando em conta o desempenho das caches, vislumbramos que o desempenho do sistema possa ser melhorado sem diminuir a satisfação dos usuários com relação a natureza do conteúdo sugerido, e deixamos a análise empírica dessa constatação como assunto para trabalhos futuros.

7. Trabalhos Relacionados

A literatura sobre o impacto de sistemas de recomendação no desempenho de caches ainda é escassa, mas está rapidamente crescendo [Verhoeyen et al., 2012, Krishnappa et al., 2015, Chatzieleftheriou et al., 2017, Sermpezis et al., 2017].

Os primeiros a considerarem a possibilidade de configurar os sistemas de caching em função dos sistemas de recomendação foram [Verhoeyen et al., 2012]. Os autores usaram-se de simulações para mostrar as potenciais vantagens de se combinar decisões de caching e de recomendação de conteúdo. No presente trabalho, em contrapartida, usamos o mecanismo de caching para perturbar o sistema de recomendação, e consequentemente a demanda dos usuários.

Em [Munaro et al., 2015] os autores estudam como sistemas de recomendação afetam o desempenho de sistemas par-a-par. Em sistemas par-a-par, quanto maior a demanda por um certo conteúdo, maior o grau de colaboração entre os pares, e menores os custos para o servidor. Similarmente, quanto maior a demanda por conteúdos populares em um sistema de cache, maior o número de *hits*. Embora existam semelhanças entre os dois trabalhos, as funções de custo e utilidade a serem maximizadas são distintas, dadas as diferenças entre sistemas de cache e sistemas par-a-par.

Recentemente, em [Chatzieleftheriou et al., 2017, Sermpezis et al., 2017] os autores propõem o estudo da relação entre sistemas de recomendação e caches. Embora a motivação de [Chatzieleftheriou et al., 2017, Sermpezis et al., 2017] seja similar a deste trabalho, a metodologia adotada é distinta. Em [Chatzieleftheriou et al., 2017, Sermpezis et al., 2017] os autores consideram uma métrica de distorção do sistema de recomendação distinta daquela considerada neste trabalho, e no presente trabalho consideramos o uso de TTLs ou RCs como modelo para caches, enquanto que os autores de [Chatzieleftheriou et al., 2017] consideram caches LRU.

Em [Krishnappa et al., 2015] os autores estudam os efeitos do sistema de recomendação no desempenho do Youtube, e usam dados reais para avaliar diferentes políticas de recomendação. No presente trabalho, em contrapartida, consideramos modelos analíticos para estudar os desafios em questão. Na linha de trabalhos analíticos que visam melhor compreender o funcionamento de redes de cache, destacamos [Kaafar et al., 2013, Adomavicius et al., 2012, Dernbach et al., 2016,

Leconte et al., 2016, Gill et al., 2007]. Embora todos estes trabalhos tenham como extensão natural o estudo da relação entre sistemas de recomendação e o desempenho de caches, nenhum considerou a demanda dos usuários como sendo a variável de controle.

8. Conclusão

Sistemas de recomendação [Gomez-Uribe e Hunt, 2016] e caches [Nygren et al., 2010] afetam milhões de usuários diariamente na Internet. Estas duas componentes tem papéis complementares no atendimento ao usuário que acessa um conteúdo: uma influencia a demanda, e a outra, influencia o atendimento dessa demanda. Fica evidente então a necessidade de melhor compreender as relações entre estas áreas, de forma a tornar possível explorar conjuntamente o potencial de ambas no atendimento ao usuário tanto no sentido de lhe prover o conteúdo mais adequado ao seu interesse, quanto a que o conteúdo seja provido de forma rápida e com boa qualidade.

Neste artigo, propomos modelos e problemas de otimização para melhor compreender a relação entre sistemas de recomendação e o desempenho de algoritmos de caching. Estendendo o *framework* de otimização proposto em [Dehghan et al., 2016], consideramos a demanda por conteúdo como sendo uma variável elástica. O papel do algoritmo de recomendação passa a ser o de controlar a demanda, de forma a melhorar os desempenho das caches, mas sem comprometer os gostos dos usuários. Mostramos que algumas instâncias do problema proposto admitem solução simples em fórmula fechada, enquanto que outras requerem solução numérica. Finalmente, apresentamos resultados preliminares usando *traces* do Movielens.

Consideramos que nosso trabalho é um passo para uma melhor compreensão as implicações mútuas da recomendação e dos algoritmos de caching, o que acreditamos ser uma frente a ser explorada no sentido de beneficiar provedores e consumidores de conteúdo, dado o potencial de aumento da QoS e também da QoE no longo prazo. Como trabalhos futuros, pretendemos avaliar experimentalmente como variações da QoE impactam as preferências do usuário, bem como incorporar no modelo a probabilidade de um usuário aceitar a recomendação oferecida pelo sistema, levando em conta que esta não é a recomendação original baseada apenas no desejo do usuário pelo conteúdo.

Referências

Adomavicius, G., Bockstedt, J., Curley, S. P., e Zhang, J. (2012). Effects of online recommendations on consumers' willingness to pay. In *Decisions@ RecSys*, pages 40–45.

Amazon, E. (2015). Amazon web services. *Available in: http://aws. amazon. com/es/ec2/(November 2012)*.

Boyd, S. e Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Chatzieleftheriou, L. E., Karaliopoulos, M., e Koutsopo, I. (2017). Caching-aware recommendations: Nudging user preferences towards better caching performance. In *INFOCOM*.

Che, H., Tung, Y., e Wang, Z. (2009). Hierarchical web caching systems: Modeling, design and experimental results. In *JSAC*.

Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., e Tay, Y. (2016). A utility optimization approach to network cache design. In *INFOCOM*.

Dernbach, S., Taft, N., Kurose, J., Weinsberg, U., Diot, C., e Ashkan, A. (2016). Cache content-selection policies for streaming video services. In *INFOCOM*, pages 1–9.

Domingues, G., e Silva, E. d. S., Leão, R. M., Menasché, D. S., e Towsley, D. (2017). Enabling opportunistic search and placement in cache networks. *Computer Networks*, 119:17–34.

Domingues, G., Leao, R. M., Menasche, D. S., et al. (2015). Flexible content placement in cache networks using reinforced counters. *arXiv preprint arXiv:1501.03446*.

Fagin, R. (1977). Asymptotic miss ratios over independent references. *Journal of Computer and System Sciences*, 14(2):222–250.

Fofack, N. C., Dehghan, M., Towsley, D., Badov, M., e Goeckel, D. L. (2014). On the performance of general cache networks. In *VALUETOOLS*, pages 106–113.

Gill, P., Arlitt, M., Li, Z., e Mahanti, A. (2007). Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM.

Gomez-Uribe, C. A. e Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13.

Kaafar, M. A., Berkovsky, S., e Donnet, B. (2013). On the potential of recommendation technologies for efficient content delivery networks. *ACM SIGCOMM Computer Communication Review*, 43(3):74–77.

Krishnappa, D. K., Zink, M., Griwodz, C., e Halvorsen, P. (2015). Cache-centric video recommendation: an approach to improve the efficiency of youtube caches. *ACM Trans. on Multimedia Computing, Communications, and Applications (TOMM)*, 11(4):48.

Leconte, M., Paschos, G., Gkatzikis, L., Draief, M., Vassilaras, S., e Chouvardas, S. (2016). Placing dynamic content in caches with small population. In *INFOCOM*, pages 1–9.

Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., e Riedl, J. (2003). Movielens unplugged: experiences with an occasionally connected recommender system. In *Proc. of the 8th international conference on Intelligent user interfaces*, pages 263–266. ACM.

Munaro, D., Delgado, C., e Menasché, D. S. (2015). Content recommendation and service costs in swarming systems. In *ICC*, pages 5878–5883. IEEE.

Nygren, E., Sitaraman, R. K., e Sun, J. (2010). The akamai network: a platform for high-performance internet applications. *Operating Systems Review*, 44(3):2–19.

Sermpezis, P., Spyropoulos, T., Vigneri, L., e Giannakas, T. (2017). Femto-caching with soft cache hits: Improving performance through recommendation and delivery of related content. *arXiv preprint arXiv:1702.04943*.

Turnbull, D. (2017). High-quality recommendation systems with elastic search. `https://dzone.com/articles/high-quality-recommendation-systems-with-elastic-2`.

Verhoeyen, M., Vriendt, J. D., e Vleeschauwer, D. D. (2012). Optimizing for video storage networking with recommender systems. *Bell Labs Tech.*, 16(4):97–113.