Sobre o Equilíbrio entre Desempenho e Justiça em Ambientes de Cache Multi-Locatário

João Ramalho[1], Ana Nery[1], Anna Lira[1], Thiago Emmanuel Pereira[1], Francisco Vilar Brasileiro[1], Mariana Mendes[2]

[1]Universidade Federal de Campina Grande
Unidade Acadêmica de Sistemas e Computação
Av. Aprígio Veloso, s/n, Campina Grande, PB, 58.429-900, Brasil

[2]VTEX
Centro Empresarial, Praia de Botafogo, 300 - 3º Andar - Botafogo
Rio de Janeiro - RJ, 22250-040, Brasil

{joao.ramalho,ana.nery,anna.lira}@ccc.ufcg.edu.br

{temmanuel,fubica}@computacao.ufcg.edu.br

mariana.mendes@vtex.com

Abstract. *Multi-tenant web caches are susceptible to performance interference, where one tenant's activity can negatively impact others. Traditional fairness policies, like FairShare, aim to mitigate this by providing fair resource allocation. However, their suitability for caching environments is not well established. In this work, we investigate the performance of FairShare using production traces from a large-scale e-commerce platform. Through simulation, we evaluate its effectiveness in balancing fairness and overall system performance. Our results reveal that while FairShare eliminates interference, it significantly degrades the cache Hit Ratio, demonstrating that classical fairness policies are not directly applicable to caching scenarios. This highlights the need for cache-aware fairness mechanisms that consider the specific characteristics of web caches.*

Resumo. *Em caches web multi-locatários, a atividade de um locatário pode prejudicar o desempenho dos demais. Políticas de alocação de recursos tradicionais, como FairShare, buscam mitigar isso alocando recursos de forma justa. No entanto, sua adequação para ambientes de cache não é conhecida. Neste trabalho, investigamos o desempenho do FairShare usando simulações de rastros de produção de uma plataforma de comércio eletrônico. Nossos resultados revelam que o FairShare não consegue equilibrar justiça e desempenho; embora o FairShare elimine a interferência, degrada significativamente o Hit Ratio do cache, demonstrando que políticas de justiça clássicas não são diretamente aplicáveis a cenários de cache. Isso destaca a necessidade de novos mecanismos de alocação cientes das características específicas de cache.*

1. Introdução

A gerência de caches multi-locatários não é simples. As cargas dos locatários não apresentam o mesmo padrão, variando de formas diferentes ao longo do tempo. Novos locatários podem se juntar ao sistema a qualquer momento, enquanto outros saem. Em

particular, essas dificuldades são exacerbadas em plataformas de comércio eletrônico. Nestas plataformas, há locatários (lojas clientes da plataforma) com diferentes volumes de requisições. Há locatários com diferentes catálogos de produtos (ou seja, a quantidade de itens em cache varia conforme o locatário), indo de poucos itens até centenas de milhares. Ainda, há diferenças nos padrões de acesso aos itens em cache de cada loja. Alguns com sazonalidade diária, outros com carga constante, ainda alguns com picos de demanda imprevisíveis [Lira et al. 2024 A, Lira et al. 2024 B].

Ainda que não seja simples operar sistemas desse tipo, é possível tirar proveito desta complexidade. Por exemplo, plataformas multi-locatários podem adotar sistemas de cache compartilhados e multiplexar os recursos entre os locatários no tempo. Assim, quando um locatário está em um momento de baixa demanda, os recursos podem ser utilizados por outros locatários. Ao fim, usa-se menos recursos para atender à demanda agregada. No entanto, apesar destes ganhos de eficiência, o compartilhamento também apresenta desafios. Em momentos de excesso de demanda, alguns locatários podem interferir no desempenho dos demais, ao removerem itens destes para que os seus sejam armazenados. Esses itens removidos, caso sejam requisitados novamente, não estarão mais disponíveis, diminuindo assim o desempenho dos locatários que sofrem interferência.

Uma alternativa para mitigar a interferência é dividir o espaço de armazenamento do cache compartilhado em partições dedicadas a cada locatário. Essa abordagem garante que a inserção de itens por um locatário não cause a remoção de itens de outros. No entanto, a definição do tamanho das partições exige cautela; uma divisão igualitária raramente é ideal, pois alguns locatários podem receber capacidade além do que podem tirar proveito, enquanto outros sofrem com recursos insuficientes para um desempenho adequado. Em resumo, embora o particionamento elimine a interferência entre locatários, pode resultar em uma redução no *Hit Ratio* global do sistema. Note que a queda na eficiência de um cache impacta não apenas as aplicações e usuários que o acessam diretamente, mas também os custos operacionais do sistema de origem (a fonte original dos dados que o cache armazena); os sistemas de origem precisarão ser dimensionados para atender a uma quantidade de requisições maior.

Neste trabalho, descrevemos os resultados de uma cooperação com a empresa VTEX[1] focada em reduzir esse tipo de interferência no cache *Web* de sua plataforma de comércio eletrônico. Devido ao modelo de negócio da empresa, que não considerar diferentes classes, é importante evitar que alguns clientes possam ser mais penalizados que outros. Para obter esse equilíbrio, podemos levar em conta critérios de justiça de acordo com as necessidades de cada locatário e os benefícios que poderão obter para a capacidade alocada. Políticas para alocação justa têm sido aplicadas em muitos domínios. Por exemplo, políticas baseadas na estratégia *Max-Min* [Bertsekas e Gallager 1996] foram aplicadas em protocolos de controle de fluxo [Hahne 1991]. Avaliamos, por simulação, o impacto no desempenho da adoção da política *Fair Share*[2] [Boudec 2002], que adota a estratégia *Max-Min*, para decisão do tamanho das partições em comparação com a estratégia atual da empresa, na qual os locatários compartilham a capacidade do cache sem reservas.

Observamos que políticas clássicas que seguem a estratégia *Max-Min* têm um

[1]https://vtex.com

[2]https://github.com/ufcg-lsd/max-min-algorithms

impacto considerável no desempenho do sistema: enquanto o *hit ratio* agregado dos locatários no cenário compartilhado atinge quase 60%, não passa de 40% quando o cache é particionado[3].

O restante do artigo está organizado da seguinte maneira: na próxima seção, apresentamos o modelo de simulação, os dados utilizados na avaliação e o algoritmo de particionamento. Em seguida, discutimos os resultados obtidos. Por fim, concluímos o artigo destacando nossas principais observações e sugerindo possíveis estratégias para aprimorar a eficiência das abordagens *Max-Min* no contexto específico de sistemas de cache.

2. Metodologia

Em nossa avaliação, consideramos que as políticas são aplicadas periodicamente para decidir o tamanho da partição de cada locatário, seguindo um critério de justiça que equilibre as necessidades de cada um deles. Consideramos a política *Fair Share* [Boudec 2002] que segue a estratégia *Max-Min*. Essa estratégia prioriza os agentes que teriam suas necessidades atendidas com a menor quantidade de recursos [Rawls 1971]. Em resumo, seguindo a estratégia *Max-Min*, não se deve aumentar a porção de recursos atribuída a um agente se isso reduzir a porção de um agente que tenha recebido uma porção menor.

Além da política *Fair Share*, consideramos duas outras estratégias: *No Partition* e *Equal Partition*. Na estratégia *No Partition*, a capacidade do cache é compartilhada entre os locatários, sem particionamento. A porção que cada locatário irá ter depende exclusivamente de sua carga e da política de reposição de itens adotada no cache. Essa estratégia reflete o cenário da VTEX e da indústria, tipicamente. Por sua vez, a política *Equal Partition* divide a capacidade do cache em porções de mesmo tamanho para os locatários. Como a demanda dos locatários não é uniforme, a política *Equal Partition* pode ser entendida como um limite inferior de desempenho entre as três políticas consideradas.

Para avaliar e comparar as políticas *Fair Share*, *No Partition* e *Equal Share*, construímos um simulador determinístico que reproduz rastros de requisições. O simulador implementa o algoritmo de reposição *Least Recently Used* (`LRU`) e permite variar a capacidade do cache e das partições (para o caso da política *Fair Share*). Utilizamos um rastro de requisições coletado do cache do serviço de catálogo de produtos da VTEX, correspondendo a um período contínuo de 10 horas de requisições, com uma amostragem de 10%. O total de requisições é superior a 56 milhões, provenientes de cerca de 5 mil locatários [Lira et al. 2024 A, Lira et al. 2024 B]. Em nossa avaliação, variamos a capacidade do cache para entender o comportamento das políticas em diferentes cenários de oferta. As capacidades foram definidas como uma fração do *Footprint*, a soma dos bytes de itens únicos acessados em uma janela de tempo [Xiang et al. 2013]. Em nosso estudo, usamos toda a duração do rastro (10 horas) no cálculo do *Footprint*.

Para o caso da política *FairShare*, a alocação de recursos dos locatários muda ao longo do tempo. Em nossas simulações, uma decisão de alocação permanece válida por janelas de 60 minutos seguintes à decisão. O processo de decisão da política *FairShare* considera a demanda de cada locatário. Consideramos a demanda futura, também em uma janela de 60 minutos, posterior ao momento de tomada de decisão. Em nossa

[3]Avaliamos também outras políticas *Max-Min*, tais como WaterFilling e MMS, que levam a resultados semelhantes, e foram omitidas da discussão por limitação de espaço.

avaliação, essa demanda futura é calculada com base nas requisições registradas no rastro de utilização. A demanda é decidida como sendo a **capacidade ótima** para o locatário, que corresponde à menor capacidade para atingir o maior desempenho (em termos de *Hit Ratio*) de cada locatário. Essa capacidade ótima é calculada usando o algoritmo de Mattson [Mattson et al. 1970]. Desse modo, por nossas simulações usarem um oráculo da demanda futura, podemos entender os resultados da política *FairShare* como um melhor caso de aplicação dessa política.

Para analisar os cenários simulados, além do Hit Ratio, observamos a **Interferência** causada e sofrida pelos locatários. Definimos Interferência quando um item é removido do cache por outro locatário e posteriormente retorna ao cache. A Interferência causada refere-se aos locatários que provocam essas remoções, enquanto a sofrida são os locatários cujos itens foram removidos. Ao serem removidos, esses itens perdem a chance de gerar um "*hit*", afetando o desempenho do locatário.

3. Resultados

A Figura 1 apresenta a distribuição acumulada do percentual da interferência causada pelos locatários, considerando simulações com a política *No Partition* (para as demais políticas, por haver particionamento, não há interferência), para caches com três capacidades diferentes (1%, 10% e 50% do *Footprint* agregado de todos os locatários). Essa distribuição é mostrada de maneira ranqueada por locatário, considerando o *Footprint* do maior para o menor. Podemos observar que a fonte de interferência é bastante concentrada em um conjunto pequeno dos maiores locatários. Por exemplo, os 500 maiores locatários (10% do total) são responsáveis por cerca de 75% de todas as requisições que causam interferência em ambos os cenários simulados. De maneira análoga, isso indica que um conjunto grande dos menores locatários, mesmo considerando-os de modo conjunto, não causa grande interferência.

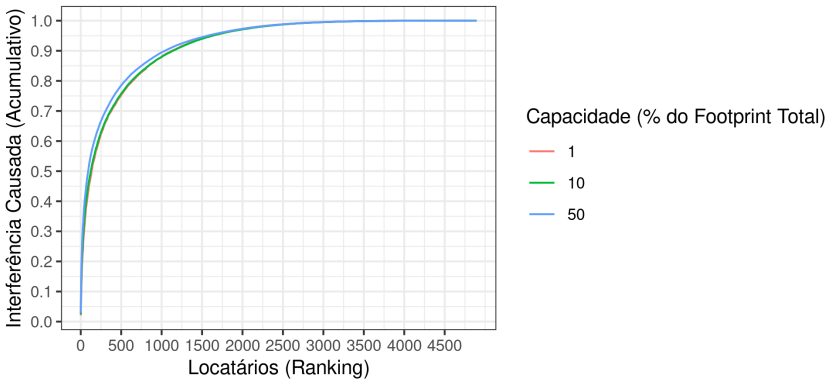


Figura 1. Percentual cumulativo das requisições que geraram interferência para cada locatário ranqueados por *Footprint* para diferentes capacidades.

A Figura 2 apresenta a distribuição da interferência sofrida pelos 40 maiores locatários (considerando *Footprint* como métrica de ranqueamento), considerando simulações com a política *No Partition* com diferentes valores de capacidade total. Primeiro, notamos que o *Footprint* por si só não explica quanto um locatário sofrerá de interferência. Por exemplo, o trigésimo quinto maior locatário sofre bem menos interferência que o primeiro, mesmo acessando um volume de dados bem menor. Outra consideração

importante diz respeito ao efeito da capacidade do cache na interferência. Alguns locatários, por exemplo, o $13º$ e o $30º$ maior deles, são bastante sensíveis ao aumento da capacidade. Por outro lado, outros locatários parecem ser insensíveis a essas variações (o $6º$ e o $35º$, por exemplo). Isso pode indicar que a alocação de capacidade, quando houver particionamento, precisa ser feita de maneira criteriosa, considerando características particulares de cada locatário.

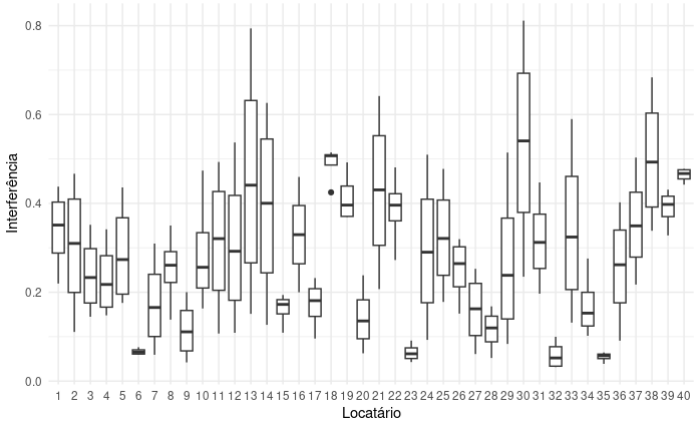


Figura 2. Interferência dos 40 maiores locatários, em Footprint, para diferentes capacidades.

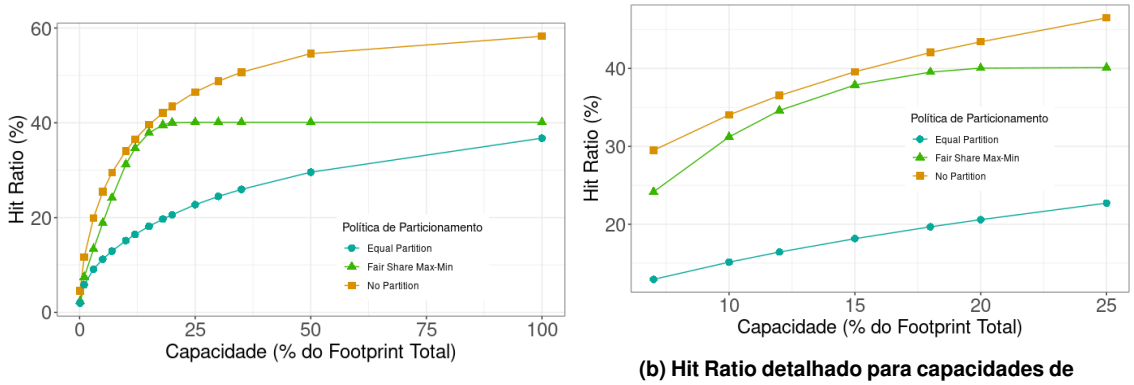
Na Figura 3a, observamos o *Hit Ratio*, para as três estratégias de gerenciamento: *No Partition*, *FairShare* e *Equal Partition*. Como esperado, a estratégia *No Partition* apresenta melhor desempenho agregado. Isso acontece porque locatários que são capazes de provocar grande interferência conseguem aumentar seu *Hit Ratio* em detrimento dos demais locatários. Note que há disparidade na frequência de acesso aos itens; ou seja, é possível que um item bastante acessado, que pode aumentar o *Hit Ratio* agregado, expulse itens com frequência baixa de acesso (esse comportamento é o que se espera de um cache que adota uma política de reposição de itens eficaz).

Por sua vez, as políticas *Equal Partition* e *FairShare* eliminam a interferência, mas o fazem com grande impacto no desempenho. A política *Equal Partition*, de modo esperado, tem o pior desempenho: chega a reduzir o *Hit Ratio* em 20 pontos percentuais em relação à política *FairShare*. A política *FairShare*, aplicada com sucesso em outros contextos, não apresenta bons resultados no contexto de cache. Embora seja melhor que *Equal Partition*, ainda fica distante dos resultados obtidos com a política *No partition* tanto nos casos de escassez severa quanto moderada, reduzindo o *Hit Ratio* 6 pontos percentuais.

4. Conclusão

A análise da interferência sofrida pelos locatários mostra que a ausência de uma política de particionamento leva a uma distribuição desigual e injusta dos recursos, beneficiando alguns locatários enquanto outros são prejudicados. Esse problema tem impacto significativo em caches compartilhados, especialmente em sistemas com locatários de características heterogêneas, como o que observamos na VTEX.

Este trabalho explorou a utilização de política de particionamento em sistemas de cache multi-locatário, utilizando uma variação do princípio Max-Min Fairness. O



(a) Hit Ratio para todas as capacidades.

(b) Hit Ratio detalhado para capacidades de 7% a 25%.

Figura 3. Hit Ratio sob diferentes capacidades utilizando diferentes políticas de particionamento.

algoritmo avaliado, apesar de eliminar a interferência entre locatários, apresentou um desempenho significativamente inferior ao ideal.

Os dados coletados e a metodologia utilizada fornecem uma base para futuras investigações, especialmente para refinar a política baseada no *Max-Min Fainess* analisada e avaliar sua viabilidade em sistemas de produção. Além disso, consideramos também a exploração de novas definições de demandas (por exemplo, taxa de repetição de itens). Essa mudança poderia capturar melhor as necessidades dos locatários, permitindo um gerenciamento de recursos ainda mais eficiente e justo.

5. Agradecimentos

Este trabalho foi financiado pela VTEX BRASIL (EMBRAPII PCEE-2304.0229).

Referências

BERTSEKAS, D.; GALLAGER, R. *Data Networks*. second. [S.l.]: Prentice Hall, 1996.

BOUDEC, J.-Y. L. Rate adaptation, congestion control and fairness: A tutorial. 01 2002.

HAHNE, E. L. Round-robin scheduling for max-min fairness in data networks. *IEEE J. Sel. Areas Commun.*, v. 9, n. 7, p. 1024–1039, 1991.

LIRA, A. et al. No clash on cache: Observations from a multi-tenant ecommerce platform. In: *Proceedings of the 2024 ACM/SPEC International Conference on Performance Engineering*. [S.l.: s.n.], 2024 A.

LIRA, A. et al. Desafios na gerência de cache web multi-locatários. In: *Anais Estendidos do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.: s.n.], 2024 B. p. 283–288. ISSN 2177-9384.

MATTSON, R. et al. Evaluation techniques for storage hierarchies. *IBM Systems Journal*, v. 9, n. 2, p. 78–117, 1970.

RAWLS, J. *A Theory of Justice*. Cambridge, Mass: The Belknap press of Harvard University Press, 1971. Eleventh printing, 1981.

XIANG, X. et al. HOTL: a higher order theory of locality. In: *ASPLOS 2013*. [S.l.]: ACM, 2013. p. 343–356.