Estratégias de Cache para a Redução do Consumo de Banda e dos Atrasos na Distribuição de Vídeo Sob-Demanda na Internet

Josilene A. Moreira[1], Márcio Neves[2], Victor Souza[3], Djamel Sadok[2]

[1]Departamento de Informática – Universidade Federal da Paraíba (UFPB)
CCEN - João Pessoa - PB - CEP 58.051-900 - Brasil

[2]Grupo de Pesquisa em Redes e Telecomunicações - UFPE - Av. Prof. Moraes Rego, S/N - Cidade Universitária - CEP 50732-970 - Recife - PE - Brasil

[3]Ericsson Research, Kista, Suécia

josilene@di.ufpb.br,
{mmn2,jamel}@gprt.ufpe.br,victor.souza@ericsson.com

Abstract. *Applications of Video-on-Demand has been widely used on the Internet in recent years, accounting for 40% of the total traffic in 2010. However, the videos are still attended with delay, which affects the quality of user experience (QoE). In addition, very large objects cause a serious impact on the profile of Internet traffic, creating congestion and consuming too much bandwidth. This paper proposes caching strategies that reduce bandwidth consumption and reduce the initial delay when compared with some well-established techniques in the literature. Using traces of YouTube videos the proposed techniques are able to improve the byte hit ratio performance in 102% even using very small caches.*

Resumo. *Aplicações de Vídeo sob-Demanda têm sido largamente utilizadas na Internet nos últimos anos, representando cerca de 40% do total de tráfego em 2010. Entretanto, os vídeos continuam sendo assistidos com atrasos, o que afeta a qualidade de experiência dos usuários (QoE). Além disso, os objetos muito grandes provocam um sério impacto no perfil da tráfego da Internet, gerando congestionamentos e alto consumo de banda. Este trabalho propõe estratégias de cache que reduzem o consumo de banda e diminuem os atrasos iniciais quando comparadas com algumas técnicas bem estabelecidas na literatura. Usando traces de vídeos do YouTube, as técnicas são capazes de melhorar o acerto em bytes em 102% mesmo usando caches muito pequenas.*

1. Introdução

As aplicações multimídia têm sido largamente utilizadas na Internet nos últimos anos, destacando-se especialmente o uso de vídeo ao vivo (*Live streaming*) e também de vídeo sob-demanda (*Video on-Demand - VoD*). Impulsionado por diversos tipos de aplicações, o crescimento do uso de vídeo sob-demanda tem se destacado, e diversas formas de VoD encontram-se hoje disponíveis. Em um estudo recente sobre as características de tráfego da Internet, a CISCO mostrou que as aplicações de VoD foram responsáveis por 40% de todo o tráfego em 2010, e essa percentagem deverá crescer para 57% em 2014 [7].

Os vídeos encontram-se disponíveis nas mais diferentes formas na Internet. Entre os principais serviços que impulsionam a utilização de VoD estão os vídeos produzidos pelos próprios usuários (*User Generated Content - UGC*). O *YouTube* é o mais bem sucedido serviço de compartilhamento de VoD no mundo, onde 65.000 vídeos produzidos pelos próprios usuários são adicionados todos os dias; estima-se que o YouTube possua mais de 100 milhões de vídeos disponíveis [31]. Na Coréia, o serviço *Daum UCC* serve dois milhões de visitantes e recebe mais de 35.000 acessos semanais, sendo conhecido por prover vídeos de alta qualidade (até 800 Kb/s) [15].

Uma outra categoria de VoD que também tem crescido destacadamente provê uma biblioteca de vídeos on-line que podem ser acessados a qualquer momento por meio de uma assinatura mensal de custo razoavelmente baixo. Exemplos destes serviços são o *NetFlix*, muito popular nos Estados Unidos, o qual oferece séries de TV e filmes e custa 7,99 dólares por mês, e o *LoveFilm* o maior serviço de VoD da Europa, que oferece um acervo de filmes com uma duração média de 94 minutos [15]. Na China, o *PowerInfo* lidera este mercado, provendo às maiores cidades chinesas um serviço de assinatura para acesso aos vídeos que conta com mais de 1,5 milhão de usuários [18].

Os vídeos de notícias são outra categoria também com grande quantidade de acessos. Disponíveis em portais como BBC, CNN, UOL, Terra e outros, as notícias recentes em vídeo atraem os usuários que em geral já dispõem de maior largura de banda. Enquanto a popularidade do acervo de vídeos on-line nos serviços por assinatura pode ser controlado de certa maneira por campanhas de marketing e apresenta um padrão de acesso relativamente previsível, o mesmo não acontece com os sites de notícias, cujo padrão de acesso é bem mais difícil de prever [2].

Entretanto, distintamente da tradicional TV, a qual foi projetada para um uso determinado e onde o conteúdo digital pode ser facilmente assistido, a Internet não foi projetada com a finalidade específica de prover vídeos. Enquanto cresce a popularidade do conteúdo UGC, dos filmes e séries e dos sites de notícias na Internet (entre outros), o consumo de largura de banda aumenta significativamente, gerando um forte impacto sobre as características de tráfego e causando diversas dificuldades tanto aos provedores de serviço como aos usuários. No modelo tradicional de distribuição de conteúdo da Internet geralmente os servidores, que são a fonte primária do conteúdo, encontram-se distantes e a transferência de dados causa alto consumo de banda. Uma vez que o caminho entre o servidor e os clientes pode ser muito longo e freqüentemente congestionado, os atrasos são imprevisíveis, prejudicando a qualidade percebida pelo usuário final. Dessa forma, a distribuição de vídeo na Internet ainda enfrenta dificuldades diversas envolvendo fatores como alto consumo de banda, latência, qualidade de serviço, infra-estrutura inadequada, baixa qualidade de experiência do usuário (*Quality of Experience - QoE*) e alto custo de distribuição [23] [30].

Uma das principais soluções que procuram amenizar estes problemas é o uso de *Caches*. Caches são basicamente servidores réplicas que guardam uma cópia do conteúdo disponível no servidor principal, e têm a finalidade de minimizar a carga de acesso no servidor, diminuir o congestionamento entre a origem e o destino dos dados e aumentar a velocidade de acesso dos usuários [11]. As técnicas de cache minimizam estes problemas trazendo o conteúdo para locais mais próximos dos clientes [14].

No entanto, as técnicas de cache para conteúdo Web não são adequadas para distribuição de vídeo. Objetos de vídeo são muito grandes quando comparados a objetos compostos por textos e imagens. Enquanto os objetos de páginas Web tradicionais são medidos em KBytes, os objetos de vídeo são medidos em MBytes. Em 2007 o tamanho médio de um objeto de vídeo na Internet era de 63 MB, 201 vezes maior do que o tamanho de uma página Web [27]. Quando estes grandes objetos de vídeo são armazenados juntamente com os objetos menores, uma grande quantidade de objetos pequenos precisam ser removidos da cache para acomodar os maiores, o que diminui a taxa de acertos no cache. Percebe-se então que os algoritmos de gerenciamento da cache apropriados para objetos Web não são apropriados para lidar com objetos de vídeo; assim, novos algoritmos de cache para vídeo precisam ser desenvolvidos [17].

Pesquisas recentes mostram o ganho potencial de um melhor uso das técnicas de cache neste cenário de uso intensivo de aplicações de vídeo na Internet [5][9][15]. Estes trabalhos mostram que 10% dos vídeos mais populares são responsáveis por cerca de 90% de todo o tráfego. Dessa forma, um mecanismo de cache que armazene os objetos mais populares pode reduzir a carga no servidor em 75% [5]. Os algoritmos procuram manter na cache os segmentos dos vídeos mais populares, a fim de proporcionar uma maior taxa de acertos e conseqüentemente diminuir o consumo de banda. O melhor desempenho de um algoritmo de cache depende então da sua capacidade de gerenciar o seu conteúdo, ou seja, definir quais são os vídeos 'corretos' a serem armazenados [11].

Este trabalho apresenta técnicas de cache desenvolvidas a fim de melhorar o seu desempenho, diminuindo o consumo de banda e reduzindo o número de vídeos que irão experimentar atraso inicial. As abordagens *PCCA - Proxy Cache Conservative Approach* e *PCAA - Proxy Cache Agressive Approach* foram desenvolvidas a partir da combinação de estratégias de segmentação dos objetos com políticas de inserção e remoção baseadas em funções que atribuem prioridades aos objetos de acordo com a sua importância e com a probabilidade de requisições futuras. As técnicas desenvolvidas são avaliadas através de comparações com várias abordagens bem conhecidas da literatura, utilizando diversos cenários que modelam as características de tráfego de vídeo da Internet. Além disso, dados reais de traces do YouTube também são avaliados.

As abordagens propostas obtêm um melhor desempenho em diversos cenários estudados, usando simulações tanto com dados sintéticos como reais. Com os traces reais do YouTube o desempenho do cache no que diz respeito à economia de banda é bastante significativo, aumentando o acerto em bytes em até 102% mesmo com caches muito pequenas. As abordagens também proporcionam a redução do atraso inicial semelhante ao desempenho de técnicas bem estabelecidas na literatura, e proporcionam um melhor equilíbrio entre as métricas de acerto em bytes e atraso inicial. Esta característica é importante porque geralmente um algoritmo só é capaz de melhorar uma das duas métricas.

2. Técnicas de Cache

Objetos de vídeo são tipicamente muito grandes e representam uma carga pesada tanto para as caches como para a rede de distribuição. Armazenar o conteúdo completos dos vídeos pode rapidamente esgotar a capacidade de uma cache. Assim, as abordagens dividem os objetos em segmentos menores e procuram gerenciar como manter os

segmentos dos objetos mais populares armazenados na cache. Esta técnica básica é conhecida como cache parcial (*Partial caching*), e a maioria das técnicas atuais continuam ainda utilizam esta estratégia. O cache parcial é freqüentemente combinado com estratégias de freqüência de acessos (*LFU*), periodicidade de acessos (*LRU*) e funções que atribuem escores de importância aos objetos (*function-based strategies*), a fim de obter abordagens ainda mais eficientes [19][20].

2.1 Trabalhos Relacionados

Em um trabalho seminal Townsley et al. [22] propõem a técnica de particionar os objetos em duas partes, prefixo e restante do objeto, conservando na cache os prefixos dos objetos mais populares (*Prefix Caching*) Na próxima requisição para o mesmo objeto, a parte inicial é transmitida ao usuário enquanto o restante do vídeo é obtido do servidor principal. Esta abordagem consegue melhorar a qualidade percebida pelo usuário, o qual experimentará menos atrasos iniciais (*start-up delays*).

Outras técnicas estendem ou modificam a abordagem de prefixo. Miao e Ortega [16] discutem as vantagens de armazenar outras partes importantes dos objetos, além dos segmentos iniciais, se houver espaço disponível na cache. A seleção dos segmentos considera o tamanho do buffer do usuário e as propriedades dos objetos de vídeo. Na abordagem conhecida por *Exponential Caching* ou *Pyramid* [28], os autores reservam uma área inicial da cache para os segmentos iniciais dos objetos e aplicam uma segmentação variável ao restante do vídeo. O tamanho do segmento aumenta exponencialmente, e o *i*-ésimo segmento tem o tamanho de 2^{i-1}.

Na técnica S*kyscraper* o tamanho dos segmentos cresce mais lentamente: o segmento *i+1* é do mesmo tamanho ou o dobro do segmento *i* [29]. Ambas melhoram significativamente o desempenho da cache em termos de atraso inicial. No entanto, a eficiência destas técnicas dependem da definição adequada dos parâmetros *tamanho do segmento inicial do vídeo* e *tamanho da porção da cache a ser reservada para guardar os segmentos iniciais*, o que representa uma desvantagem. Se estes parâmetros não forem bem ajustados, a eficiência da cache pode ser reduzida. Usamos a técnica *Pyramid* na nossa avaliação de desempenho, para medir a eficiência dos nossos algoritmos em termos de redução da métrica atraso inicial.

Uma importante abordagem conhecida como *Lazy* foi proposta por Chen et al. [4]. Neste método, todos os objetos são inseridos completamente no cache na primeira requisição, e a decisão sobre o tamanho dos segmentos é adiada para o momento em que um objeto precisa ser removido da cache. Lazy utiliza uma função para atribuir importância aos objetos, a qual tenta prever os acessos futuros aos objetos baseada na freqüência e duração dos acessos já realizados. Alcança um bom desempenho em termos de acertos em bytes (*byte hit*) e conseqüente diminuição do consumo de banda, e é uma das abordagens consideradas importantes para gerenciamento de caches. Usamos esta técnica como um *baseline* para efeitos de comparação com relação à métrica de *byte hit*.

Em [8] os autores analisam o impacto da granularidade dos segmentos sobre o desempenho da cache, observando que a granularidade pode ser ajustada para proporcionar o aumento do *byte hit*. O uso de segmentos menores conduz a um melhor desempenho quando a popularidade dos objetos é relativamente estável, enquanto que

em períodos de variações bruscas de popularidade, o uso de segmentos maiores melhora o desempenho em termos de *byte hit*.

Wei et al. apresentam duas propostas, os algoritmos PAPA (*Popularity-Aware Partial Caching Algorithm for VoD*) [24] e DECA (*Dynamic Segment-based Caching Algorithm)* [26] onde cada segmento é subdividido em prefixo e sufixo, e os prefixos dos objetos mais populares são todos armazenados na cache. Dessa forma, quando um usuário realiza uma operação de adiantar o vídeo, os prefixos dos pontos-âncora são encontrados enquanto a cache busca o restante do servidor de origem. No algoritmo DECA, o tamanho dos segmentos é ajustado de acordo com a popularidade do objeto.

Satsiou and Paterakis [21] combinam a arquitetura das caches em dois níveis com as técnicas de segmentação. O primeiro nível de caches está localizado mais próximo do servidor e contém os segmentos iniciais dos objetos, enquanto que as caches de segundo nível estão mais distantes e guardam os segmentos restantes. As caches do primeiro nível usam a abordagem LRU (*Least Recently Used*), enquanto as caches do tipo B usam a abordagem LRLFU, proposta para capturar simultaneamente os aspectos de freqüência e antiguidade dos objetos.

O trabalho apresentado em [6] constata que o número e a capacidade das caches combinados com a estratégia de gerenciamento dos objetos afetam grandemente a performance do sistema de distribuição. Se as caches são muito grandes, podem atrair um número muito grande de requisições e desequilibrar o sistema; por outro lado, se são muito pequenas, acabam por não ajudar a reduzir a carga total no servidor principal. Na nossa avaliação de desempenho estudamos o impacto do tamanho da cache variando a sua capacidade de 5 a 70% do tamanho total dos objetos presentes no sistema [4][26].

2.2 Técnicas propostas

Utilizamos duas métricas principais para medir a eficiência das estratégias de cache, que são a taxa de acertos em bytes (*byte hit ratio*) e o atraso inicial (*start up delay*). Enquanto o *byte hit* deve ser maximizado, consequentemente reduzindo o consumo de banda na rede, o *start up delay* precisa ser minimizado, melhorando a qualidade de experiência do usuário (QoE). Em geral, as técnicas de cache ou diminuem o consumo de banda ou melhoram a experiência do usuário, mas dificilmente atingem os dois objetivos simultaneamente.

Este trabalho apresenta uma estratégia de cache com duas variações compostas por políticas de armazenamento de prefixo, de segmentação de substituição de objetos no cache. A primeira variação, *Proxy Cache Conservative Approach (PCCA)*, obtém o melhor equilíbrio na redução concomitante do consumo de banda e diminuição do atraso inicial. A segunda variação, *Proxy Cache Aggressive Approach (PCAA),* é mais agressiva e alcança o objetivo de aumentar a taxa de acertos em bytes.

Proxy Cache Conservative Approach (PCCA)

Segmentação

Um tamanho adequado do prefixo pode ajudar a melhorar o desempenho do algoritmo de cache. Assim, a segmentação adotada por PCCA utiliza um tamanho variável para o segmento inicial (prefixo), o qual depende do tamanho do vídeo. O primeiro segmento

tem 10% do tamanho total do objeto, o segundo segmento tem o dobro do tamanho do primeiro, e um tamanho exponencial do segmento é adotado a partir daí. Usando esta abordagem criamos uma segmentação exponencial dinâmica que varia de acordo com o tamanho do vídeo e reduz o problema causado pela necessidade de ajuste dos parâmetros presente na abordagem exponencial apresentada em [28].

Política de Admissão de objetos

Quando há espaço livre na cache para armazenar o objeto completo, este é armazenado. A partir do momento que a área de armazenamento da cache torna-se saturada, a política de admissão começa a ser aplicada. Nesta fase, quando uma requisição é feita para um objeto que não está na cache, o seu prefixo será sempre armazenado; os outros segmentos só poderão ser armazenados a partir da próxima requisição para o mesmo objeto. Com essa estratégia, é possível manter um maior número de prefixos na cache, diminuindo o número de erros ao requisitar um objeto (*cache miss*).

Política de substituição de objetos

A política de substituição usa uma função de prioridade para selecionar os objetos que serão retirados da cache. Quando não há espaço livre, a função é computada para todos os objetos presentes na cache que não estão sendo acessados no momento: o objeto com menor valor de prioridade é escolhido. O último segmento do objeto escolhido é removido da cache e o procedimento é repetido até que haja espaço livre para armazenar o novo segmento. Entretanto, um prefixo só pode ser retirado para armazenar outro prefixo, o que ajuda a manter os prefixos dos objetos mais populares na cache e diminui o número de objetos que experimentam atraso inicial.

A função de prioridade de cache (CP) considera vários aspectos dos objetos de vídeos como a freqüência de acessos em um período de tempo, o número total de acessos e a probabilidade de acessos futuros para atribuir um valor de prioridade para um determinado objeto. Estes atributos são então mantidos em uma estrutura de dados para cada um dos objetos na cache (Equação 1).

Equação 1

$$CP = \frac{n}{(T_c - T_r)} * MIN\left\{1, \frac{\frac{T_r - T_1}{n}}{T_c - T_r}\right\}$$

Equação 1 – Função de Prioridade da Cache

n = número de acessos do objeto

T_c = timestamp atual

T_1 = timestamp do primeiro acesso ao objeto

T_r = timestamp do último acesso ao objeto

$\frac{n}{(T_c - T_r)}$ = número médio de acessos em um intervalo de tempo

$MIN\left\{1, \frac{\frac{T_r - T_1}{n}}{T_c - T_r}\right\}$ = probabilidade de acessos futuros

A probabilidade de acessos futuros é baseada em [3] e baseia-se no número de acessos e no momento dos acessos. A função usa $T_c - T_r$ que representa o intervalo entre o timestamp atual e o timestamp do último acesso ao objeto. $\frac{T_r - T_1}{n}$ representa o intervalo médio entre os acessos que aconteceram para um determinado objeto. Assim, se $T_c - T_r < \frac{T_r - T_1}{n}$, estima-se que existe uma grande probabilidade que aconteçam novos acessos ao mesmo objeto em breve; caso contrário, a probabilidade de acessos futuros é considerada pequena e a prioridade do objeto ser mantido na cache diminui.

Quando existe espaço livre para armazenar o segmento requisitado, ele é sempre armazenado. Se não houver espaço, duas situações podem ocorrer. Caso seja a primeira requisição, o prefixo será sempre armazenado, e a função de prioridade será calculada para escolher o objeto a ser removido. Qualquer segmento poderá ser removido para armazenar o prefixo do novo objeto. Quando o objeto a ser retirado é selecionado, seus segmentos são removidos a começar do último, até que haja espaço suficiente. Se ainda assim não houver espaço suficiente, outro objeto será então removido usando o mesmo critério. Se não houver espaço suficiente no cache mas o objeto já tiver sido requisitado anteriormente, a política de remoção é invocada para liberar espaço para o novo segmento até que haja espaço suficiente. Lembrando que, neste caso, o prefixo do objeto escolhido nunca será retirado para acomodar segmentos do novo objeto que não sejam o prefixo. Resumindo, um prefixo só poderá ser substituído por outro prefixo.

Proxy Cache Aggressive Approach (PCAA)

Nesta abordagem a segmentação do objeto é semelhante ao PCCA. Entretanto, duas políticas diferentes são aplicadas para modificar a política de substituição, as quais impactam na taxa de acerto em bytes e conseqüentemente reduzem o consumo de banda. Primeiro, qualquer vídeo será inteiramente armazenado na primeira vez que for requisitado. Isto é, enquanto usando PCCA apenas o prefixo do vídeo será armazenado na primeira requisição, usando PCAA todo o vídeo sempre será inserido na cache na primeira requisição. A segunda modificação é que, enquanto usando PCCA o primeiro segmento do vídeo só poderá ser substituído por outro primeiro segmento, esta restrição não se aplica à abordagem mais agressiva. Assim, qualquer parte do vídeo poderá ser retirada para que um novo vídeo possa ser inserido completamente.

3. Metodologia

É utilizada uma simulação de um conjunto de requisições feitas a uma cache. Considera-se que a distância entre o servidor de origem dos dados e a cache é grande e assim cada vez que o objeto requisitado não está na cache, os dados precisam ser baixados do servidor central, incorrendo em consumo de banda. A distância entre o cache e os clientes é desprezível. Para efeito de comparação, cada conjunto de requisições é processado simulando-se as estratégias de cache propostas e também as estratégias Exponencial, Lazy e LRLFU.

A seleção da carga de dados a ser usada em uma simulação é uma parte extremamente importante da avaliação de desempenho, que pode contribuir para revelar o comportamento das técnicas estudadas [12]. São modelados diversos cenários com

características de tráfego típicas da Internet e, além disso, são utilizados traces de dados reais de acessos a vídeos do YouTube.

3.1 Carga de Dados Sintéticos

As características de tráfego das cargas de dados sintéticas estão resumidas na Tabela 1.

Tabela 1 - Características de tráfego das cargas de dados sintéticos

Característica	Valor	Comentário
Número de requisições	10.000	Baixo número de requisições
	50.000	Alto número de requisições
Número de vídeos	10% do número de requisições	Pequena quantidade de vídeos
	40% do número de requisições	Alta quantidade de vídeos
Tamanho do vídeo	10 MB	Vídeos pequenos
	600 MB	Vídeos grandes
Distribuição de popularidade	Zipf(α=0.4)	Distribuída (*spread*)
	Zipf(α=0.8)	Concentrada (*skewed*)
Percentual de objetos *cacheable*	30%	Percentagem de objetos requisitados mais de uma vez
	50%	
Duração das sessões de usuários	Vídeo completo	Sessão longa
	80% dos usuários assistem 20% dos vídeos, 20% assistem o vídeo completo	Sessão curta

A fim de obter uma avaliação de desempenho detalhada, os cenários foram construídos com a variação de todos os parâmetros mostrados na Tabela 1. A chegada das requisições segue uma distribuição de Poisson (λ=1/30). Considera-se que a cache é capaz de atender a todas as requisições que chegam. O gerador de carga de dados sintéticos ProWGen [1] foi usado para construção dos diferentes cenários, o qual permite a especificação de parâmetros como a distribuição Zipf, o número de requisições e a percentagem de objetos *cacheable* (objetos requisitados mais de uma vez). O tamanho dos vídeos pequenos foi definido com uma média de 10 MB (como no YouTube) e os vídeos grandes com 600 MB. A duração das sessões foi definida como curta, onde 80% dos usuários assistem 20% dos vídeos, e longa, onde o vídeo é assistido completamente [25]. São modeladas duas distribuições de popularidade típicas, uma onde a popularidade é mais distribuída entre os objetos, e outra onde a popularidade é mais concentrada em um menor número de objetos [3][13]. O número de objetos presente no sistema também influencia no desempenho do cache [15]. Assim, preparamos cargas de dados com pequena e grande quantidade de vídeos.

As principais métricas utilizadas são o percentual de acertos em bytes (*byte hit ratio*), relacionado com o consumo de banda, o percentual de vídeos com atraso inicial (*delayed start*), relacionado com a qualidade de experiência do usuário e a performance balanceada, que procura mostrar qual das abordagens proporciona um desempenho equilibrado entre as duas métricas básicas. Foram realizados experimentos com todas as combinações de fatores e níveis mostrados na Tabela 1. Cinco algoritmos foram avaliados em cada execução, PCCA, PCAA, *Lazy*, *Pyramid* e LRLFU. Estes três últimos foram escolhidos por obterem excelente desempenho em termos de *byte hit* (*Lazy*) e *delayed start* (*Pyramid* e LRLFU). O tamanho da cache é um fator importante

para o desempenho do algoritmo; assim o tamanho da cache variou de 5 a 70% do tamanho total dos objetos presentes no sistema[3][4][10].

3.2 Dados Reais

Os dados reais usados nas simulações foram obtidos a partir de uma coleta de traces do YouTube realizada por [32], em um ponto de acesso localizado entre uma universidade e a Internet no período de Junho de 2007 a Maio de 2008. Os dados contém os traces das requisições dos clientes e dos vídeos requisitados. As principais características destes traces encontram-se na Tabela 2.

Tabela 2 – Características dos traces do YouTube

Característica	Trace A 031508	Trace B 012908.24	Trace C 091507.24	Trace D 031208
Número de requisições	12838	18750	46407	64113
Número de vídeos	9548	13584	30229	41881
Percentual de tráfego *cacheable*	21%	19%	25%	25%
% de objetos distintos	74%	72%	65%	64%

Os *timestamps* das requisições originais foram usados nas simulações, assim como o tamanho real dos objetos. O tamanho médio dos vídeos foi de 8.4 MB, e a popularidade próxima a uma distribuição Zipf com α=0.8. Como esperado, observa-se uma baixa percentagem de vídeos *cacheable*, típica de sistemas como o YouTube, onde um grande número de vídeos é introduzido no sistema a cada dia.

4. Resultados

4.1. Cenários Sintéticos

O primeiro estudo de caso avalia um cenário sintético formado por 50.000 requisições dirigidas a um grande número de objetos, 20.000. A percentagem de objetos *cacheable* é de 30% e a popularidade dos objetos segue uma distribuição Zipf com parâmetro α=0.8, significando que a popularidade está concentrada em um número relativamente pequeno de objetos. Os vídeos utilizados têm um tamanho médio de 10 MB (vídeos pequenos).

De acordo com análises de tráfego presentes em pesquisas recentes, esta configuração é muito representativa de um cenário do YouTube [9][15][32]. Uma característica marcante do serviço do YouTube é de apresentar um número muito grande de objetos disponíveis no sistema, já que milhares de vídeos são enviados todos os dias. ESte cenário possui 20.000 objetos, correspondendo a 40% do total de requisições. Os vídeos são assistidos completamente, isto é, as sessões não são interrompidas.

A Figura 1 mostra o *byte hit* alcançado pelas técnicas propostas comparado com as abordagens LRLFU, Pyramid e Lazy. Lembramos que a técnica Lazy é usada como baseline para avaliar o desempenho em termos de *byte hit*, enquanto Pyramid e LRLFU são usadas para comparação do desempenho em termos de atraso inicial. A abordagem PCAA obtém o melhor desempenho entre todos os algoritmos, para todos os tamanhos de cache. A performance dos algoritmos em geral cresce quando o tamanho da cache aumenta. Para um tamanho de cache de 5%, PCAA consegue um desempenho 22% melhor do que Lazy, o que representa uma grande economia de banda mesmo usando caches pequenas.

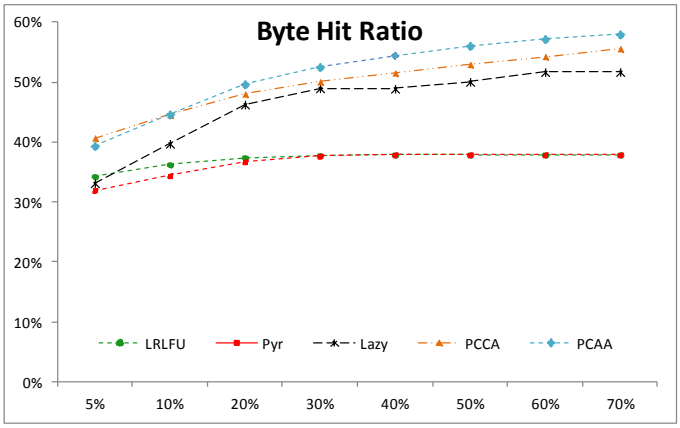

Figura 1 - Acertos em bytes para vídeos pequenos

Como os dois algoritmos inserem o vídeo completo na primeira requisição, a diferença em performance provavelmente deve-se ao esquema de segmentação juntamente com a função de prioridade, as quais são diferentes nas duas abordagens.

Observa-se ainda que o desempenho da estratégia mais conservadora é semelhante ao da mais agressiva, principalmente para caches pequenas. As estratégias LRLFU e Pyramid obtém baixo desempenho em termos de *byte hit* para todos os tamanhos de cache. Entretanto, estes dois algoritmos obtém o melhor desempenho na redução do atraso inicial (Figura 2).

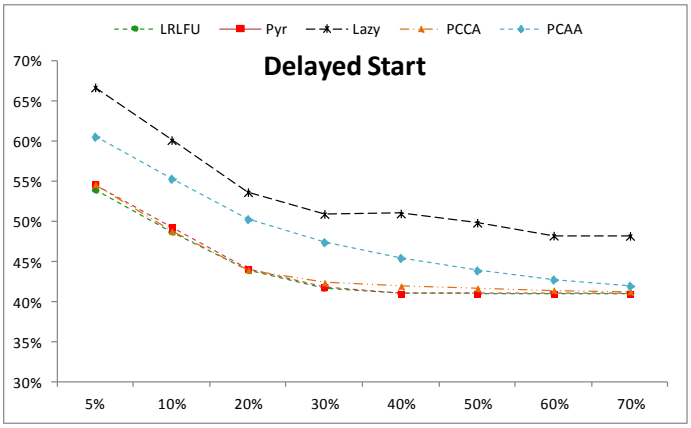

Figura 2 - Atraso inicial para vídeos pequenos

Percebe-se que a técnica mais conservadora PCCA é capaz de igualar o desempenho de Pyramid e LRLFU. Este é um excelente resultado, visto que PCCA não reserva um espaço inicial para os primeiros segmentos dos vídeos, mas mesmo assim consegue reduzir a quantidade de vídeos com atraso inicial.

As abordagens Lazy e PCAA obtêm os piores desempenhos; mesmo assim, PCAA ainda consegue um resultado melhor do que Lazy, alcançando uma performance similar aos melhores resultados quando a cache aumenta de tamanho (a partir de 50%). Como as técnicas PCAA e Lazy inserem todo o objeto na primeira requisição, a capacidade da cache esgota-se rapidamente. Assim, um número menor de objetos é encontrado na cache, causando uma degradação na eficiência medida em termos de atraso inicial.

Podemos observar que as técnicas que obtém o melhor desempenho na taxa de acerto em bytes não conseguem o melhor desempenho em termos de redução do atraso inicial. A Figura 3 mostra a performance balanceada para este cenário. Em geral, uma técnica que melhora o *byte hit* piora o atraso inicial. Entretanto, percebe-se que o método PCCA provê um melhor equilíbrio entre as duas métricas para todos os tamanhos de cache. A segunda abordagem com melhor desempenho é a PCAA.

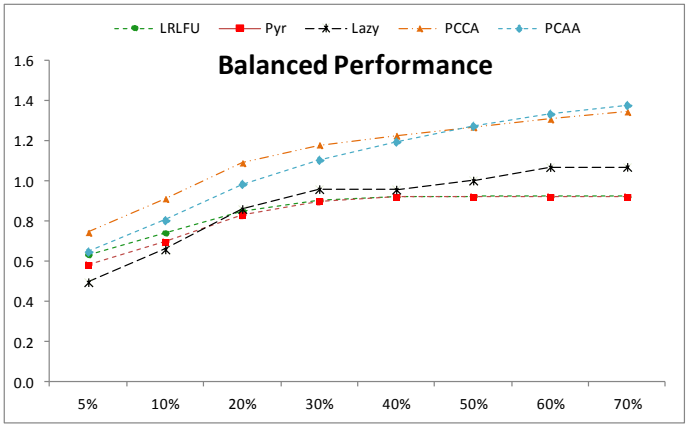

Figura 3 - Performance balanceada para vídeos pequenos

4.2. Cenários Reais

Por questões de espaço mostraremos apenas os resultados obtidos na avaliação dos traces C e D, os quais possuem um grande número de requisições e um grande número de objetos. Podemos considerar que a avaliação destes traces provê uma boa visão sobre o comportamento das abordagens em serviços de vídeos produzidos pelos usuários.

Analisando-se a taxa de acertos em bytes, verifica-se que a abordagem PCAA obtém o melhor desempenho principalmente para tamanhos de cache bem pequenos. Para uma cache de 5%, PCAA apresenta um desempenho 102% melhor do que a técnica Lazy. Mesmo quando a cache tem um tamanho de 10%, o desempenho ainda continua 56% melhor (Figura 4). A partir de um tamanho de cache de 30%, a abordagem Lazy iguala o desempenho da PCAA. Verifica-se ainda que nas técnicas LRLFU e Pyramid a convergência ocorre para uma cache menor, alcançando 10% de acertos em bytes.

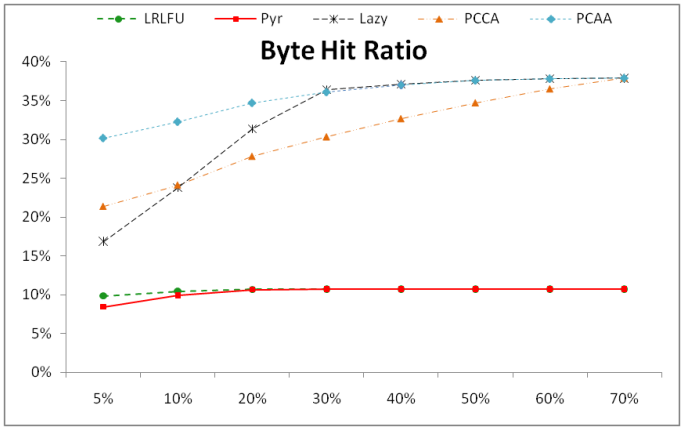

Figura 4 - Acerto em bytes para o trace C do YouTube

A Figura 5 corrobora os achados anteriores, e mostra que a performance é ainda melhor com um maior número de requisições. Este é um resultado excelente, obtido a

partir de dados reais de um cenário típico do YouTube. Torna-se difícil, na prática, re-dimensionar constantemente a cache em relação ao número de objetos disponível no sistema. Em geral, as caches permanecem com um tamanho fixo por um determinado período de tempo e, quando comparadas ao número de objetos disponíveis no YouTube, são quase sempre pequenas. A nossa análise revela que, mesmo com cache bem pequenas, o desempenho do algoritmo PCAA é bem superior ao desempenho das outras estratégias. Sendo assim, concluímos que a técnica pode realmente contribuir para diminuir o consumo de banda na rede, em serviços como o YouTube. As Figuras 6 e 7 mostram a percentagem de atraso inicial alcançado pelas técnicas nos traces C e D.

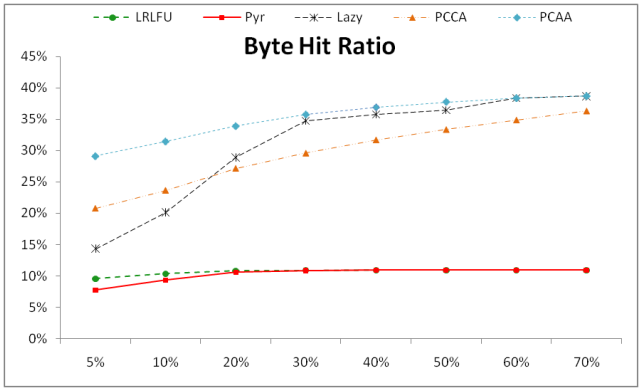

Figura 5 - Acerto em bytes para o trace D do YouTube

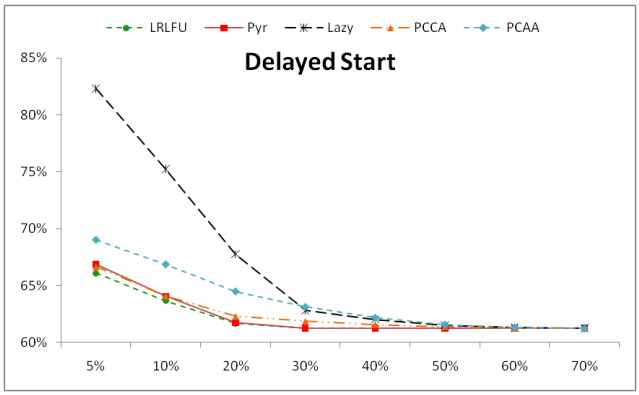

Figura 6 – Atraso inicial para o trace C do YouTube

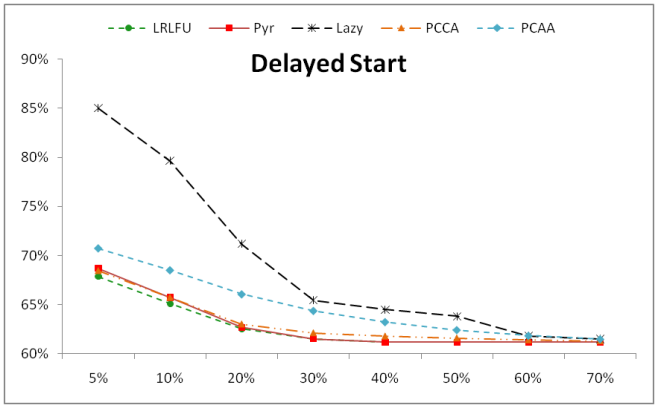

Figura 7 – Atraso inicial para o trace D do YouTube

5. Conclusões

Os algoritmos de cache são extremamente importantes e estão diretamente relacionados com o desempenho das caches tanto em termos de consumo de banda como em termos de redução de atrasos. Através de diversos experimentos usando dados reais e sintéticos, mostramos que a técnica de cache PCAA proposta é capaz de alcançar um desempenho significativamente melhor em termos de acerto em bytes do que técnicas conhecidas na literatura. Também mostramos que a técnica PCCA consegue proporcionar um melhor equilíbrio entre o consumo de banda e os atrasos iniciais. Além disso, PCCA consegue reduzir o percentual de objetos com atraso inicial, apresentando um desempenho semelhante ao de técnicas bem estabelecidas na literatura. Destacamos que os melhores resultados foram obtidos utilizando traces reais do YouTube, o que comprova que o uso de mecanismos de cache adequados é capaz de afetar tanto o consumo de banda na rede mundial como melhorar a experiência do usuário.

Referências

[1] Busari, M.; Wiliamson, C., "ProWGen: A Synthetic Workload Generation Tool for Simulation evaluation of Web Proxy Caches", Journal of Comp. Networks, 2002.

[2] Buyya, R., Pathan, M. and Vakali, A. (eds.), "Content Delivery Networks", ISBN: 978-3-540-77886-8, Springer, Berlin, Germany, 2008.

[3] Chen, S., Shen, B., Wee, S., and Zhang, X. "Adaptive and Lazy segmentation based proxy caching for streaming media delivery". In Proceedings of the 13th international Workshop on NOSSDAV, Monterey, CA, USA, June 2003.

[4] Chen, S., Wang, H., Zhang, X., Shen, B., and Wee, S., "Segment-Based Proxy Caching for Internet Streaming Media Delivery". IEEE MultiMedia 12, July 2005.

[5] Cheng, X. Dale, C.,Liu, J.,"Statistics and Social Networks of YouTube Videos", IWQoS 2008. 16th International Workshop on In Quality of Service, 2008. pp. 229-238.

[6] Cholvi, V. and Segarra, J. "Analysis and placement of storage capacity in large distributed video servers". Computer Communications 31, September 2008.

[7] CISCO, "Cisco Visual Networking Index: Forecast and Methodology, 2009-2014". 2010, June. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf. Accessed in November, 2010.

[8] Elias Balafoutis, Antonis Panagakis, Nikolaos Laoutaris, and Ioannis Stavrakakis. "Study of the Impact of Replacement Granularity and Associated Strategies on Video Caching". In Cluster Computing 8, pages 89-100, January 2005.

[9] Gill, P., Arlitt, M., Li, Z., Mahanti, A. "Youtube traffic characterization: a view from the edge". In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement* (IMC '07). ACM, New York, NY, USA, 15-28.

[10] Hefeeda, M.; Saleh, O., "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems" in IEEE/ACM Transactions on Networking, 2008.

[11] Hofmann, M., and Beaumont, L. R., "Content Networking: Architecture, Protocols, and Practice". Morgan Kaufmann Publishers, San Francisco, CA, USA, 2005.

[12] Jain, Raj. "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley- Interscience, New York, NY, April 1991, ISBN:0471503361.

[13] Jayarekha, P., P. C. Air, T. R. Gopalakrishnan, “A Rank Based Replacement Policy for Multimedia Server Cache Using Zipf-Like Law”, Journal of Computing Vol 2, Issue 3, March 2010.

[14] Akamai Swoops on Red Swoosh”. Light Reading, April 12, 2007. http://www.lightreading.com/document.asp?doc_id=121710. Acesso em Nov, 2009.

[15] M. Cha , H. Kwak , P Rodriguez , Yong-Yeol Ahn , Sue Moon, “I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System”, Proc. of the 7th IMC 2007, San Diego, California, USA, October 2007.

[16] Miao, Z. and Ortega, A. 1999. Proxy Caching for Efficient Video Services over the Internet. In Proceedings of the 9th International Packet Video Workshop (PVW'99).

[17] Podlipnig, Stefan and Boszomenyi, L. A survey of Web cache replacement strategies. ACM Computer Survey, 374,398. December 2003. Dezembro, 2003.

[18] PowerInfo. http://www.sjdd.com.cn/english/products.htm. Acessado in Nov, 2010.

[19] Rodrigues, M., Neves, M., Moreira, J. Sadok, D., Callado, A., Karlsson, P., Souza, Victor, “On Traffic Locality and QoE in Hybrid CDN-P2P Networks”. In Proceedings of 44th Annual Simulation Symposium (ANSS11), SCS, Boston, USA, April, 2011.

[20] Romano, Sam, ElAarag, H. "Comparison of function based web proxy cache replacement strategies". In Proc. of 12th Intern. Conf. on Performance Evaluation of Computer & Telecomm. Systems (SPECTS'09), IEEE Press, NJ, USA, 2009.

[21] Satsiou, A., Paterakis, M., “Efficient caching of video content to an architecture of proxies according to a frequency-based cache management policy”, Proc. of the 2nd Intern. Workshop on Adv. Archit. and Algorithms for Internet Delivery, ACM, New York, 2006.

[22] Sen, S., Rexford, J., and Towsley, D. Proxy prefix caching for multimedia streams. In Proceedings of IEEE INFOCOM'99. IEEE Computer Society, 1999.

[23] Sentinelli, A., Marfia, G., Gerla, M., Kleinrock, L., “Will IPTV Ride the Peer-to Stream?”, In IEEE Communications Magazine, June 2007.

[24] Shen, L., W. Tu, and Steinbach, E., "A flexible starting point based partial caching algorithm for video on demand”, in Proc. IEEE Int. Conf. Multimedia and Expo (ICME'07), Beijing, China, Jul. 2007.

[25] Yu, H., Zheng, D., Zhao, B., Zheng W. “Understanding user behavior in large-scale video-on-demand systems”. In Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems, ACM, NY, USA, June, 2006.

[26] W. Tu, E. Steinbach, M. Muhammad, and X. Li, "Proxy Caching for Video on Demand Using Flexible Starting Point Selection”, IEEE Transactions on Multimedia, pp. 716-729, vol. 11, no.4, June 2009.

[27] Website Optimization, “Average Web Page Quintuples since 2003”. http://www.websiteoptimization.com/speed/tweak/average-web-page/ Acessado Ago, 2010.

[28] Wu, K., Yu, P. S., and Wolf, J. L. “Segment-based proxy caching of multimedia streams”. In Proc. of the 10th Intern. Conference on WWW, Hong Kong, 2001.

[29] Wu, K., Yu, P. S., and Wolf, J. L., "Segmentation of multimedia streams for proxy caching”, Multimedia, IEEE Transactions on , vol.6, no.5, pp. 770-780, Oct, 2004.

[30] Zapater, Marcio Nieblas; Bressan, Graca. “A Proposed Approach for Quality of Experience Assurance of IPTV”, In 27th Intern. Conf. Distrib. Computing Systems, (ICDCS '07), 2007

[31] Zhou, R., Khemmarat, S., Gao,L., "The Impact of YouTube Recommendation System on Video Views", in Proc. of Internet Measurement Conference (IMC), 2010.

[32] Zink, M., Suh, K., Gu, Y. and Kurose, J., "Watch Global Cache Local: YouTube Network Traces at a Campus Network - Measurements and Implications", IEEE MMCN, 2008.