UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Universidade Federal de Pernambuco

Centro de Informática

Graduação em Engenharia da Computação

Avaliação de Algoritmos de Cache Slicing para Redes de Distribuição de Conteúdo

Álefy Matheus Alves Santos

Brasil

11 de Julho de 2017

Álefy Matheus Alves Santos

Avaliação de Algoritmos de Cache Slicing para Redes de Distribuição de Conteúdo

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Universidade Federal de Pernambuco

Centro de Informática

Graduação em Engenharia da Computação

Orientador: Prof. Dr. Djamel Fawzi Hadj Sadok

Coorientador: Prof. Dr. Glauco Gonçalves

Brasil

11 de Julho de 2017

Dedico este trabalho ao meu Jesus e à minha amada família.

Agradecimentos

Agradeço primeiramente ao meu amado e querido Deus, a quem devo completamente a minha vida, a minha conquista e tudo o que me tornei. Graças à sua fidelidade e suas bençãos derramadas em minha vida pude permanecer firme, enfrentando cada dia com um coração cada vez mais grato e cheio de esperança que chegaria onde quisesse. Onde Ele quisesse.

Agradeço imensa e eternamente aos meus maravilhosos pais Aécio Alves e Virgínia Geny, e ao meu querido irmão e futuro engenheiro Victor Ádony, que estiveram sempre ao meu lado, investindo em mim e me proporcionando os melhores momentos da minha vida. À mesma medida, agradeço às minhas queridas tias, "Vovó"Lena e Dadau, que me abrigaram em seus lares, providenciando tudo do melhor para que eu pudesse fazer uma graduação mais proveitosa. Não menos importante, sou muito grato às minhas tias Dena e Moreco, que sempre demonstraram muito amor para comigo, enchendo o meu coração de alegria. E a toda a minha família, muito obrigado por estarem sempre orando por mim e me apoiando, de uma forma ou de outra. A todos esses, dedico este trabalho.

Gostaria de registrar minha gratidão a alguns dos meus professores. Primeiramente, aos professores Djamel Sadok e Glauco Gonçalves, meus orientadores, que me ajudaram imensamente em todo esse processo. E, em nome de toda a minha jornada acadêmica, agradeço ao professor Daniel Cunha, por toda a compreensão, ajuda e ensino. Com toda a certeza, tive mentores e professores maravilhosos.

Na faculdade, tive a oportunidade de conviver com vários amigos, que com certeza me ajudaram bastante na caminhada. Agradeço especialmente aos meus amigos Rennason Carneiro, Wilton Santana e Hélder Felix. Obrigado por todo esse tempo juntos, meus amigos. Sucesso a todos nós.

Também tive a oportunidade de participar do CITi, empresas júnior do Centro de Informática, na qual pude trabalhar e crescer ao lado de pessoas maravilhosas. Em especial, agradeço aos meus amigos Leonardo Silva, Estéfane Oliveira, Leonardo Coutinho, José Luciano e Sarah Cristina. Foi um tempo muito feliz com vocês.

No meu período de estágio, pude conhecer e conviver com um pessoal maravilhoso. Agradeço em especial ao meu chefe Raphael Barros, por toda a compreensão e apoio, e aos meus amigos Emanuel Silva, Hugo Leonardo, Diego Ferreira, André Alcântara, Jacinto Felipe e Tarcísio Coutinho. Foi um tempo muito bom que dividimos juntos.

Serei eternamente grato a Da. Aurora Maria e ao Sr. Adailton Costa, que me incentivaram e me deram todo o suporte possível em todo esse período, não faltando em compreensão e amor para comigo. Mais uma vez, obrigado por tudo.

Não poderia esquecer de registrar minha gratidão à toda a família da Igreja Batista do 13 de Maio, em João Pessoa. Vivi com vocês momentos inesquecíveis e aprendi muito com tudo e com todos. Aproveitando o contexto, agradeço a PIB João Pessoa e ao pessoal maravilhoso que estou tendo a oportunidade conhecer e conviver.

Por fim, quero registrar a gratidão ao meu pequeno, meu amado filho, Luiz Davi. Que Deus me permita, a partir de agora, poder ficar pra sempre perto de você e ser o melhor pai que Deus me permitir ser. Papai te ama muito!

Resumo

A Internet vem passando por grandes mudanças em virtude do aumento no tráfego da rede, dado pela adoção do acesso de banda larga e aumento no número de dispositivos que, de alguma forma, estão conectados. Além disso, o aumento na demanda por conteúdo multimídia faz com que seja necessária uma infraestrutura que obedeça aos requisitos específicos que não são completamente cobertos pela Internet. Sendo assim, vêm sendo amplamente utilizadas as Redes de Distribuição de Conteúdo (CDNs), cuja a infraestrutura é eficientemente implementada para entrega de conteúdo. Além disso, a utilização de fatiamento de cache em um ambiente *multi-tenancy* (multi-inquilino) aumenta a qualidade dos resultados obtidos. Esse trabalho, portanto, tem como objetivo avaliar algoritmos de fatiamento de cache (*cache slicing*), dadas as possíveis variações de cenário, analisando por fim o quão distantes os resultados desses algoritmos estão do ponto ótimo, alcançado por um modelo de otimização construído nesse mesmo trabalho. Os resultados mostraram que, dentre os algoritmos implementados, o algoritmo de fatiamento de cache com distribuição de fatias restantes baseada no *ranking* de *load* por *surrogate* produziu resultados que mais se aproximaram dos obtidos através do modelo de otimização. A utilização dessa estratégia mostrou-se bastante eficiente.

Palavras-chaves: CDN, fatiamento de cache, modelo analítico.

Abstract

The Internet has been undergoing major changes due to the increase in network traffic, given the adoption of broadband and increase in the number of devices that are connected in some way. Furthermore, the increase in demand for multimedia content requires an infrastructure that meets specific requirements that are not completely covered by the Internet. Therefore, Content Delivery Networks (CDNs) have been widely used, whose infrastructure is efficiently implemented for content delivery. Besides that, the use of cache slicing in a multi-tenancy environment increases the quality of the results obtained. This work, therefore, aims to evaluate cache slicing algorithms, given the possible scenario variations, analyzing at last how far the results of these algorithms are of the optimum point, reached by an optimization model constructed in the same work. The results showed that, among the algorithms implemented, the cache slicing algorithm with remaining slice distribution based on the surrogate load ranking produced results that were closer to those obtained through the optimization model. The use of this strategy proved to be quite efficient.

Keywords: CDN, cache slicing, analytical model.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CDN	Content Delivery Networks
CP	Content Provider
DHTML	Dynamic HyperText Markup Language
IaaS	Internet as a Service
IoT	Internet of Things
ISP	Internet Service Provider
LRU	Last Recently Used
MIT	Massachusetts Institute of Technology
P2P	Peer-to-Peer
PoP	Points of Presence
QoS	Quality of Service
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SP	Slice Proportion
URI	Uniform Resource Identifier
VoD	Video on Demand

Sumário

1 Introdução

Com o passar dos anos, a Internet tem experimentado um grande crescimento no tráfego de rede devido a adoção ao acesso à banda larga e também devido ao aumento no número de dispositivos conectados. A Internet conecta dispositivos inclui computadores tradicionais, dispositivos móveis, bem como dispositivos que estão inseridos no que se chama de Internet das Coisas (IoT) [1].

A Internet também tem testemunhado um aumento na complexidade do sistema e na riqueza do conteúdo. Esta complexidade pode ser atribuída ao número de aplicações que têm surgido na Internet e como elas interagem entre si. Além disso, outros fatores como aquisição, criação e distribuição de conteúdo, acabam levando a infraestrutura geral da rede aos seus limites, e em certos casos impactando no tráfego da rede, quando leva-se em conta, por exemplo, o tráfego multimídia.

A demanda por conteúdo não é fácil de ser prevista e, prover recursos para essa demanda, pode impor requisitos que poder ser difíceis de serem preenchidos no ambiente de rede. Em casos onde o número de eventos eleva-se amplamente em um determinado período de tempo, dada uma demanda alta por recursos naquele período, pode haver uma sobrecarga enorme nos servidores de Internet. Esses eventos, também conhecidos como *flash crowds*, muitas vezes resultam em indisponibilidade temporária do serviço, por não conseguir atender a alta demanda.

Procurando lidar com essas limitações da Internet, as **Redes de Entrega de Conteúdo** (*Content Delivery Networks - CDNs*) têm aparecido como uma tecnologia que serve para prover uma melhor infraestrutura e técnicas de entrega de conteúdo de uma maneira mais eficiente. Uma CDN se comporta como uma plataforma de entrega transparente e eficiente de conteúdo, com o intuito de aumentar a qualidade da experiência e do serviço.

1.1 Motivação

As CDNs têm sido bem-sucedidas na distribuição de conteúdo na Internet. Grandes empresas como Akamai [2] e Networks and Mirror Image [3] têm investido e operado com muita força ao redor do mundo. A Akamai, por exemplo, é responsável por praticamente metade desse mercado. Outros poucos competidores compõem o restante do mercado, fazendo com que seja extremamente difícil para que novos competidores construam e mantenham sua infraestrutura de CDN com condições de competir com as que já existem. Além disso, os preços apresentados por esses provedores de CDN estão longe de facilitar a entrega de conteúdo principalmente para pequenas e médias empresas, universidades, dentre outros.

Este tipo de CDN é também desenvolvida de forma estática e usa mecanismos de redirecionamento centralizados. Um dos paradigmas bastante utilizados para lidar com a provisão de recursos é adotar princípios de Computação em Nuvem, fazendo com que seja capaz prover corretamente a quantidade de recursos baseado na demanda atual [4].

Sendo assim, se mostrou conveniente utilizar recursos da nuvem com o fim de construir a chamada CDN orientada à nuvem, que aparece com o intuito de reduzir o esforço e custo de fornecer uma solução para entrega de conteúdo. Isto é feito usando a nuvem para mapear os elementos da CDN em componentes virtuais na nuvem. O paradigma se torna, então, um modelo onde usuários podem consumir recursos computacionais como um serviço e pagar pelo que eles usam. A nuvem pode ser observada como uma camada conceitual que torna transparente todos os recursos de software e hardware disponíveis e serviços que tornam possível acessar esses recursos através de uma interface bem definida [5].

Nos últimos anos, foi observado o aumento no número de provedores de armazenamento em nuvem. Podemos citar dentre eles a Amazon S3, Rackspace, dentre outros. Tais provedores operam centros de dados que podem oferecer armazenamento de conteúdo baseado em Internet e entregar eficiente com a certeza de manter a qualidade do serviço e tempo aceitável de entrega dos conteúdos [6].

Essas capacidades permitem (i) que um pequeno negócio se torne um cliente de múltiplos provedores em diferentes localizações ao redor do mundo, possibilitando a consolidação de uma cobertura dinâmica que toma vantagem dos baixos preços em locais específicos, e (ii) que se ofereça o serviço de entrega de conteúdo para terceiros, sem o custo de possuir ou operar centrais de dados distribuídos geograficamente.

A vantagem de tais modelos é que, diferente das CDNs tradicionais, em um ambiente de nuvem, há a possibilidade de construir uma cobertura com uma topologia que pode ser diferente da cobertura da rede. Atualmente, os conteúdos trafegados na Internet são bastante diversificados, onde podem também ser dinâmicos ou estáticos, agradar diferentes pessoas, levando a diferentes requisitos SLAs (*Service Level Agreements*) e diferentes algoritmos para lidar com o processo de entrega.

Neste contexto, uma CDN orientada a nuvem fornece um modelo que considera a diversidade, onde cada tipo de conteúdo ou proprietário do conteúdo pode ser lidado como uma "CDN virtual", construída no topo da nuvem com a quantidade necessária de recursos, evitando assim uma "sobre-provisão". CDNs também podem apresentar vantagens no fatiamento dos recursos da Nuvem.

1.2 Objetivos

Este trabalho tem como objetivo avaliar, de forma estática, o comportamento de uma CDN *multi-tenant*, estudando as melhores formas de se distribuir conteúdo em servidores espalhados geograficamente, por consequência distribuindo eficientemente tal conteúdo entre os clientes que utilizam o serviço.

Para alcançar o objetivo principal, foram definidos os seguintes objetivos específicos:

- Implementar um algoritmo de *cache slicing* que distribua fatias de conteúdo nos surrogates espalhados de modo autônomo;
- Utilizar parte da implementação para análise do algoritmo sem *cache slicing*, observando o seu comportamento;
- Comparar as técnicas com o fim de observar as vantagens produzidas pela técnica de *cache slicing*;
- Construir e implementar um modelo analítico, com o intuíto de comparar com os resultados dos experimentos anteriores.

1.3 Organização do trabalho

Este trabalho está dividido em 6 capítulos, incluindo este capítulo de Introdução. No Capítulo 2 serão apresentadas informações sobre os diversos sistemas de entrega de conteúdo existentes. No Capítulo 3 será dada uma atenção particular às Redes de Distribuição de Conteúdo, apresentando os tipos de arquiteturas utilizadas e as diversas categorias de CDNs existentes. No Capítulo 4, será apresentado o algoritmo de *Cache Slicing*, bem como o funcionamento de cada uma das fases desse algoritmo. Em seguida, no Capítulo 5, serão apresentados os detalhes de implementação e resultados obtidos. Por fim, o Capítulo 6 apresentará as considerações finais sobre o trabalho.

2 Sistemas de Entrega de Conteúdo

A Internet usada mundialmente por diferentes pessoas e para diferentes propósitos, tem sido cada vez mais utilizada, resultando em novos desafios. Para lidar com tais desafios, diversos modelos de cobertura de rede foram propostos e desenvolvidos, cada um com seus propósitos específicos, como CDN e P2P, para distribuir conteúdo. Alguns autores definem uma cobertura de rede como um conjunto de nós conectados por uma rede. Alguns autores [7] apresentam três coberturas de rede que vêm crescendo: Redes de Entrega de Conteúdo (CDNs), coberturas de roteamento e cobertura de segurança. Uma cobertura de roteamento existe para modificar e supervisionar o caminho dos dados na rede. Redes de segurança são focadas em prover diferentes formas de proteção à comunicação. Apesar das diferenças funcionais desses tipos de rede, elas compartilham peculiaridades que podem ser apontadas observando a definição das redes. Todas elas são formadas por um conjunto de nós interconectados e cooperam para um propósito comum. Além disso, a análise desses nós, por parte das redes, é fator chave na tomada de decisões que possam contribuir para o aumento da eficiência da rede. Por exemplo, ao analisar um nó de uma CDN, um operador poderia realocar *surrogates* ou conteúdo, dado um *flash crowd* inesperado ou simplesmente por um mal funcionamento da rede. Da mesma forma, uma rede P2P poderia re-atribuir seus super-nós para lidar com eventos de *flash crowds*. Este capítulo tem por objetivo, portanto, apresentar os modelos/sistemas de distribuição de conteúdo mais conhecidos e as principais tecnologias de rede envolvidas nesse processo.

2.1 Modelo Cliente/Servidor

A maneira mais simples de distribuir conteúdo é baseada no **modelo cliente/servidor**. Adota-se basicamente um único modelo onde um servidor ou múltiplos servidores, com cargas balanceadas (mas localizados em um mesmo lugar), são responsáveis por servir as requisições de conteúdo. Esta abordagem tem muitas desvantagens. Uma delas é que a distância entre os servidores e os usuários finais impacta na utilização dos recursos de rede, na latência e na disponibilidade. Além disso, uma infraestrutura centralizada pode também ser altamente custosa e ineficiente.

2.2 Multicast

Alguns autores [8] propuseram o uso do **modelo *multicast*** como um mecanismo de distribuição para entrega de conteúdo, considerando dois modelos de *multicast*: uma *multicast* perfeita, onde existe apenas uma fonte localizada na rede e todos os clientes formam um grupo *multicast* ao mesmo tempo; e a *multicast* realista, onde clientes se juntam de acordo

com um processo que chega, significando que tanto os grupos de associação por conteúdo como a árvore de distribuição podem mudar com o tempo. A vantagem do modelo *multicast* é superar o problema do uso de rede e carregamento do servidor, apresentados no modelo Cliente/Servidor.

2.3 Redes de Distribuição Peer-to-Peer

Uma **rede *Peer-to-Peer***, também conhecida como P2P, pode ser observada como uma arquitetura distribuída onde os participantes compartilham parte de seus recursos de hardware com o fim de prover um conjunto de serviços ou conteúdos de rede. Os recursos compartilhados pelo "ponto" são diretamente acessíveis para outros pontos sem ter que passar pelo controle de entidades intermediárias. O BitTorrent, por exemplo, é uma forma popular de uma rede de distribuição P2P. Redes P2P possuem tecnologias de histórico de comprovação, o que torna bastante eficiente na distribuição de conteúdo. No entanto, alguns requisitos de aplicações específicas, como aplicações de vídeo por demanda (VoD), tornam a distribuição P2P mais desafiadora. No caso do BitTorrent, por exemplo, a organização habilita os pontos para trocar quaisquer segmentos dos conteúdos que estão sendo distribuídos; a ordem em que os seguimentos chegam não é importante. Tal técnica não é própria para aplicações de *streaming*, por exemplo. Além do mais, em eventos de *streaming* de vídeo se utilizam técnicas para lidar com a degradação da qualidade do vídeo, ao invés de tratar os atrasados envolvidos nesse processo.

2.4 Redes de Distribuição de Conteúdo (CDNs)

As **Redes de Distribuição de Conteúdo** são redes construídas no topo da Internet, que espalham servidores de cache em diversos lugares para replicar conteúdo. Os detalhes sobre sua arquitetura, os tipos de CDN existentes, bem como os problemas que esse modelo resolve serão apresentados no capítulo seguinte.

3 Redes de Distribuição de Conteúdo (CDNs)

3.1 Definição

A CDN é uma rede de cobertura que distribui conteúdo aos *surrogates* estrategicamente localizados próximos aos usuários finais. É um conjunto colaborativo de elementos de rede em que o conteúdo é replicado com o fim de tornar a entrega eficiente e transparente. O propósito de uma CDN é melhorar a qualidade de serviço, provendo de forma rápida e eficiente a distribuição do conteúdo [9]. Isto é alcançado estabelecendo diversos **pontos de presença (PoP)** com servidores de *cache*, também chamados **surrogates** ou **servidores espelhados**, que armazenam cópias do conteúdo original e servem às requisições dos usuários em nome de um servidor de origem (**origin server**). Uma visão abstrada de uma CDN pode ser vista na Figura 1 [10].

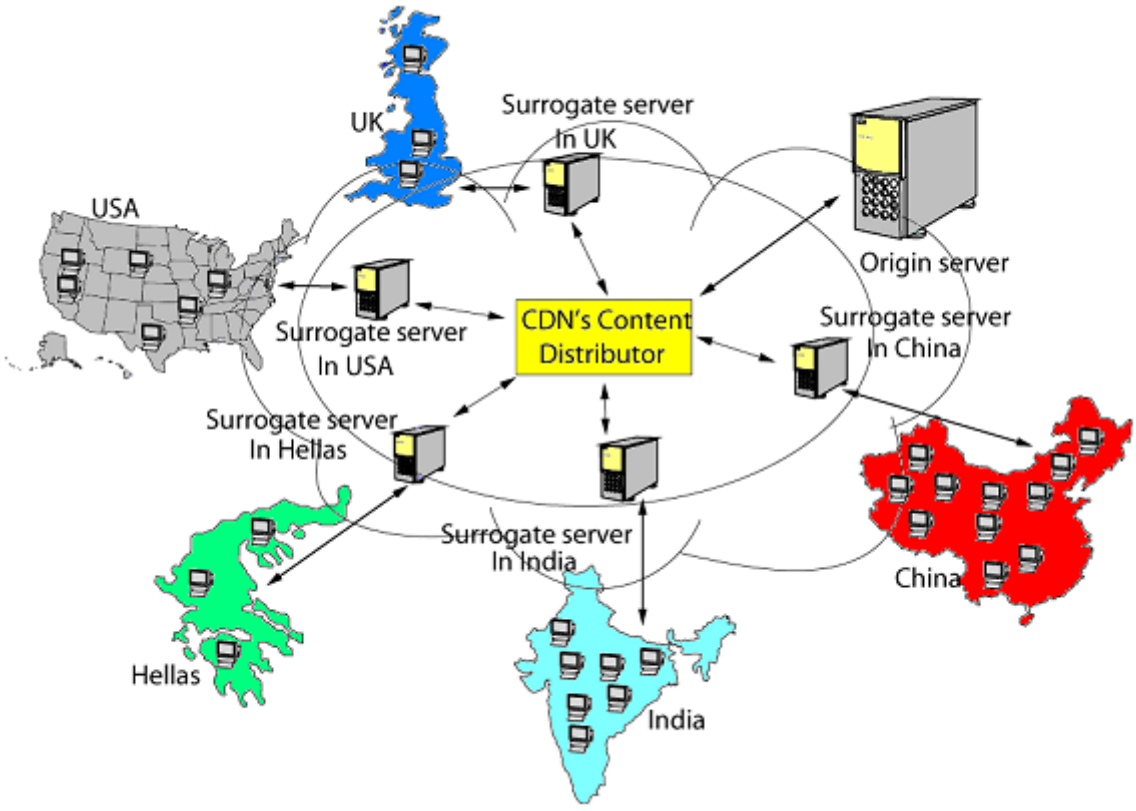


Figura 1 – Visão abstrata de uma CDN

Uma topologia consiste basicamente de um **conjunto de *surrogates*** espalhados pelas bordas da rede, aos quais as requisições são redirecionadas; **roteadores** e outros elementos

de rede; um **servidor de origem**, que mantém o conteúdo original a ser distribuído através da CDN; um **redirecionador de requisições**, que escolhe um *surrogate* apropriado que satisfaça as requisições dos usuários; um **sistema de monitoramento** e um **mecanismo de contabilização**, que provê informações e logs sobre os servidores de origem.

O conteúdo é basicamente um objeto requisitado por determinado usuário e é provido primariamente pelo servidor de origem. Este conteúdo pode ser replicado por demanda ou de antemão, onde é enviado aos *surrogates* distribuídos na rede. Uma CDN pode ser caracterizada de acordo com a sua estrutura, ou seja, de acordo com o número de *surrogates* e redirecionadores, de acordo com os protocolos envolvidos nesta operação, requisitos de serviço, dentre outros fatores. Estas configurações representam uma decisão de engenharia importante que deve ser tomada pelo provedor de CDN e impõe diversas implicações em termos de eficiência. Existem basicamente três atores principais que atuam numa CDN:

- **Provedor de conteúdo (*Content Provider*)**: representa o proprietário de conteúdo. Pode ser tanto o produtor do conteúdo como pode ser uma entidade que mantém esse conteúdo para servir os usuários. Esse provedor de conteúdo (CP) é um cliente para o provedor da CDN, que procura economizar os custos de infraestrutura e por consequência contrata um serviço de CDN para ter o seu conteúdo espalhado em diversos *surrogates*. Portanto, o CP atribui um Identificador Uniforme de Recursos aos objetos que são distribuídos. Relacionamento com os usuários finais e certos aspectos de segurança se tornam tarefas bastante importantes a serem lidadas pelos CPs.

- **Provedor de CDN**: é o proprietário da infraestrutura do conteúdo. Ele foca na construção da infraestrutura de rede (servidores e backbones de rede) para a entrega do conteúdo. Armazenamento e gerenciamento do conteúdo, isolamento dos CPs, entrega do conteúdo, são alguns exemplos de serviços e funcionalidades que o Provedor de CDN deve prover com eficiência. Ele cobra aos seus clientes (provedores de conteúdo) de acordo com a entrega de conteúdo aos usuários finais. Portanto, ele não deve apenas monitorar e coletar o uso de informações para dar suporte necessário, mas deve também lidar com operações de manutenção e expansão do serviço.

- **Usuários finais**: são os reais clientes de um provedor de conteúdo. Eles acessam em última instância os serviços providos por um provedor CDN. Tais serviços podem ser pagos ou gratuitos, sendo isso determinado pelo CP. O provedor da CDN deve ser transparente aos usuários finais. Este relacionamento com o CP inicia assinando ou acessando seus serviços. Quando um conteúdo é requisitado, o usuário é redirecionado a um dos *surrogates* da CDN.

A colaboração entre esses atores e componentes da CDN pode ocorrer em ambientes homogêneos e heterogêneos (em termos de acesso a hardware, software e capacidades de

comunicação). Por consequência, administradores de CDN devem garantir que os *surrogates* sejam dispostos em lugares estratégicos na rede. Para o provedor da CDN, a decisão de onde colocar esses *surrogates* afeta diretamente os custos de operação da própria rede. Portanto, a localização de *surrogates* é um desafio chave nas pesquisas de CDN, sendo frequentemente modelada como um problema de custo mínimo, bastante conhecido como problema dos K-médios mínimos.

3.2 Arquitetura da CDN

Os principais componentes de uma arquitetura CDN estão apresentados na Figura 2: 1. O servidor de origem, que pode ser provido tanto pelo CP como pelo servidor espelhado na infraestrutura do provedor da CDN; 2. O cliente ou usuário final; 3. Um conjunto de *surrogates*, que armazenam cópias dos conteúdos do servidor de origem e são encarregados de entregar tais conteúdos aos usuários finais; 4. Um redirecionador, ou componente de roteamento de requisições, que é responsável por direcionar as requisições aos *surrogates* apropriados; 5. Um gerenciador de conteúdo, também chamado distribuidor, que transporta o conteúdo do servidor de origem aos *surrogates* e garante a consistência do conteúdo nos *surrogates*, mantendo tais *surrogates* atualizados; 6. Um conjunto de monitores, que unem dados e informações estatísticas dos diferentes elementos chave da CDN e apresenta medidas na rede para ajudar no processo de decisão na CDN, tais como redirecionamento e distribuição de conteúdo; e 7. Um mecanismo de contabilização, que mantém os logs dos acessos dos clientes e informações sobre o uso dos servidores da CDN. Nenhuma das CDNs tem todos esses componentes. Os *surrogates*, por exemplo, podem por si mesmos ser responsáveis pela distribuição do conteúdo, evitando a necessidade de um gerenciador de conteúdo. Da mesma forma, algumas outras funcionalidades podem ser direcionadas para outros componentes, como por exemplo, o gerenciador de conteúdo pode ser também responsável pela contabilização e roteamento de requisições.

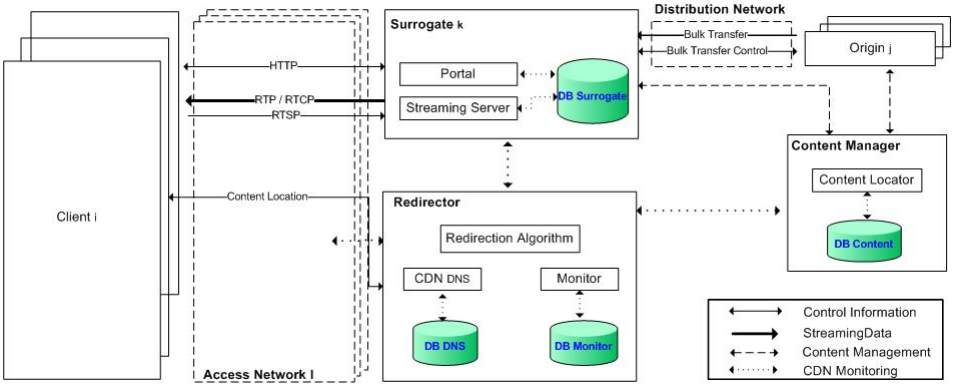


Figura 2 – Principais componentes de uma CDN

As interações entre os elementos da CDN são apresentadas na Figura 3. Tais interações

ocorrem da seguinte forma [11]:

1. O servidor de origem delega seu espaço de nome URI (*uniform resource identifier*) para ser distribuído e entregue pela CDN;

2. O servidor de origem publica o conteúdo no gerenciador de conteúdo;

3. O gerenciador de conteúdo transporta o conteúdo aos *surrogates* e atualiza a informação no sistema de roteamento de requisições, com o fim de permitir ao mesmo escolher o *surrogate* adequado;

4. O cliente requisita um objeto do qual ele acredita estar na origem, mas na verdade, a requisição é direcionada ao sistema de roteamento de requisições;

5. O roteamento de requisições escolhe o *surrogate* mais adequado para servir as requisições do usuário;

6. O *surrogate* selecionado entrega o conteúdo requisitado ao cliente e envia a informação de contabilização ao sistema de contabilização;

7. O sistema de contabilização agrega e refina a informação em detalhes do conteúdo para uso por parte do servidor de origem e da organização de cobrança. Esta informação pode também ser usada pelo gerenciador de conteúdo e pelo sistema de roteamento de requisições com o fim de prover carregamento balanceado dos conteúdos;

8. A organização de cobrança usa esses registros para cobrar ao usuário da CDN, o CP, pelo serviço de entrega.

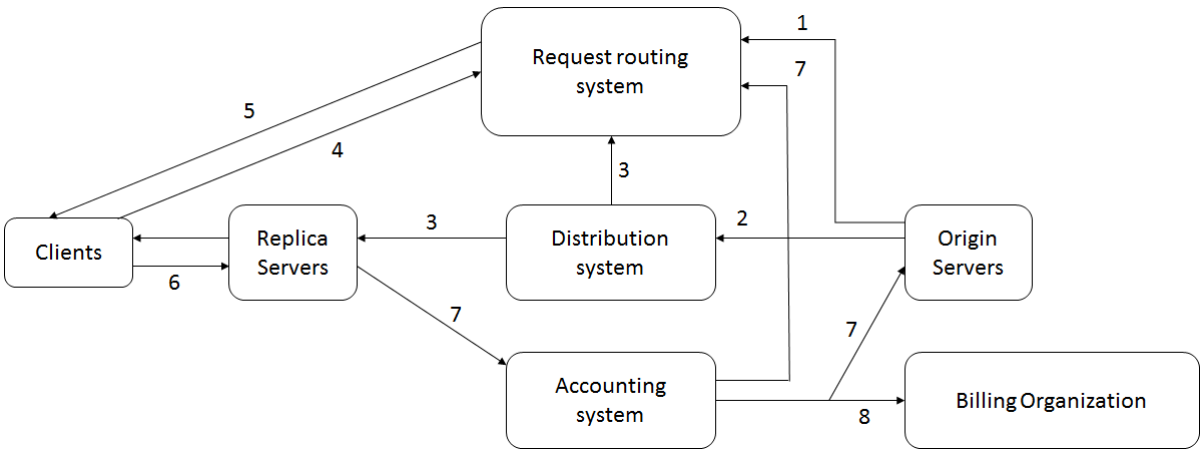


Figura 3 – Interações entre os elementos da CDN

3.2.1 Servidor de origem

O servidor de origem representa o proprietário do conteúdo atual da CDN. Ele publica o conteúdo copiando os objetos aos servidores *surrogate*. Uma questão técnica que aparece é a seleção do conteúdo a ser entregue ao usuário final pela CDN. Isto tem um grande impacto em termos de preço e de desempenho da CDN.

O conteúdo do servidor de origem pode ser replicado aos surrogates de forma total ou parcial [12]. A seleção e entrega dos conteúdos é uma abordagem simplificada onde os *surrogates* fazem a replicação dos conteúdos do servidor de origem. Isto parece ser factível quando objetos são pequenos, como por exemplo, páginas Web. No entanto, com o aumento contínuo no tamanho dos objetos, especialmente conteúdo de vídeo, e a presença de conteúdo dinâmico, faz com que o problema de armazenar e atualizar o conteúdo seja pouco gerenciável. Além disso, esta abordagem também pode resultar do uso exagerado dos recursos de rede, devido à replicação desnecessária de partes de conteúdos que são raramente requisitadas. Sobre replicação parcial, a CDN deve decidir quais objetos serão encaminhados para os *surrogates*.

O objetivo do problema de disposição das réplicas dos conteúdos é decidir em qual local tais réplicas ficariam no sistema, com o fim de **maximizar** o desempenho percebido pelo cliente, dada uma infraestrutura existente, ou minimizar o custo de infraestrutura dado o desempenho de um determinado sistema. Tais objetivos são abstraídos na definição do problema. Algumas políticas que dizem respeito às réplicas dos conteúdos foram propostas com o fim de gerenciar a replicação dos objetos e podem ser categorizadas da seguinte forma:

- **Encaminhamento baseado empiricamente**: o engenheiro da CDN decide qual conteúdo será encaminhado. Alguns autores [13] propuseram uma personalização no processo de caching dos objetos, tratando-os de forma diferente, de acordo com as políticas decididas pelo engenheiro. Heurísticas são aplicadas com o fim de se tomar uma decisão empírica. O problema dessa abordagem é devido à incerteza em escolher a heurística correta;

- **Encaminhamento baseado na popularidade**: apenas os objetos mais populares são encaminhados para os surrogates [14]. A principal desvantagem desse modelo é devido ao fato de que definir objetos populares consome bastante tempo e estatísticas confiáveis não garantidas, devido à variação da popularidade de cada objeto.

- **Encaminhamento baseado no objeto**: o conteúdo é replicado aos *surrogates* em unidades de objeto. Esta abordagem é gulosa, pois precisa escolher um objeto a ser replicado de acordo com o maior ganho de desempenho [15]. Apesar da alta complexidade de implementação, ela alcança o melhor desempenho.

- **Encaminhamento baseado em cluster**: O conteúdo é replicado em unidades de cluster. O cluster é definido como um grupo de objetos que têm algumas características em

comum. Um grupo pode ser criado estática ou dinamicamente, baseado na mudança de parâmetros, como por exemplo a carga do servidor.

3.2.2 O servidor *surrogate*

Os servidores de réplica, também conhecidos como *surrogates*, são servidores que armazenam réplicas do conteúdo existente no servidor de origem. Uma CDN é normalmente classificada de acordo com sua estrutura, número de *surrogates*, localização e algoritmo utilizado para gerenciamento do *surrogates*. Cada servidor *surrogate* tem uma base de dados que contém uma lista de todas as sessões de *streaming* disponíveis, objetos armazenados no próprio *surrogate* e informações do monitor, para gerenciamento da CDN. Normalmente, dois tipos de problemas podem aparecer para os surrogates: problemas de alocação e gerenciamento de espaço nos *surrogates*.

3.2.2.1 Problema de localização

O problema de localização ou disposição dos *surrogates* é tratado como um processo de entrega de conteúdo. Tal disposição coloca uma ênfase extra na questão de escolher o melhor local para cada *surrogate* e o número de *surrogates* necessários numa infraestrutura de rede. Algumas abordagens modelam o problema como um problema de localização central: para a disposição de um dado número de centros, eles minimizam a distância máxima entre um nó e o centro mais próximo. Algumas variantes deste problema são as k-árvores hierarquicamente separadas [16] e o problema K-centro [17]. O problema do K-centro mínimo é NP-completo, de alta complexidade e computacionalmente intensivo para ser usado na prática. Existem algumas variantes que restringem a capacidade do serviço nos centros, requerendo que cada instalação sirva um número de requisições que não ultrapasse a capacidade definida para aquele local. Devido à alta complexidade destes algoritmos, algumas estratégias de disposição de *surrogates* foram desenvolvidas:

- **Algoritmo Guloso [18]**: Esta abordagem se comporta da seguinte forma: ela escolhe M réplicas entre N potenciais lugares, escolhendo uma réplica de cada vez. Na primeira iteração, cada um dos N potenciais lugares é avaliado com o fim de determinar se é adequado aquele *surrogate* receber a réplica. O custo é computado, assumindo que todos os clientes irão acessar aquele lugar. O lugar de menor custo é escolhido. Na segunda iteração, o algoritmo busca por um segundo lugar, em conjunto com o lugar que já foi escolhido. A iteração continua até que M surrogates sejam escolhidos. O algoritmo funciona bem mesmo com entradas imperfeitas.

- **Estratégia de posicionamento baseado na topologia [19]**: este modelo considera um número fixo de máquinas candidatas onde os *mirrors* serão colocados. Um ISP (*Internet Service Provider*) pode ter um grande número de máquinas espalhadas de acordo

com a capacidade da Internet em receber os *mirrors*. O problema é em qual subconjunto de máquinas candidatas um CP deve colocar seus *mirrors*. Nesta abordagem, as réplicas são colocadas (usando o Algoritmo Guloso) nos nós com mais links de saída primeiro. É assumido que os nós com mais links de saída podem alcançar mais nós com latências menores.

- **Estratégia Hot Spot [20]**: O algoritmo *Hot Spot* tenta colocar as réplicas próximas aos clientes gerando o maior carregamento possível. Ela ordena N potenciais lugares de acordo com a quantidade de tráfego gerado em sua vizinhança. Ele coloca as réplicas nos M lugares que geram um maior tráfego. A vizinhança é definida como um círculo centrado no nó com um dado raio (número de nós distantes).

Tais algoritmos foram propostos no começo da última década, com o problema da localização de *proxies* de servidor web.

3.2.2.2 Gerenciamento de cache

Gerenciamento de conteúdo é uma tarefa essencial da CDN. Existe inclusive um debate sobre como uma CDN deve decidir onde o conteúdo deve ser colocado, usando uma abordagem centralizada ou distribuída, permitindo que os *surrogates* decidam que conteúdo deve ser replicado [9]. A primeira é a abordagem de replicação de conteúdo e foi discutida anteriormente. A outra é conhecida como abordagem de *Web caching*. Organização de cache envolve integração com outras partes da arquitetura da CDN, especialmente o gerenciador de conteúdo. Isto afeta diretamente as métricas de performance da CDN, como a taxa de acerto, latência percebida, etc.

Na abordagem de *Web caching*, o último mais recentemente usado (algoritmo LRU) é uma política de gerenciamento de cache amplamente conhecida e adotada por conta da sua eficiência, dada a sua simplicidade. Algumas técnicas de *caching* são integradas à técnica de *Web caching* com replicação de conteúdo, tornado o modelo híbrido.

O algoritmo de *caching* deve lidar com a popularidade do conteúdo e predizer a mudança da popularidade, com o fim de preparar a CDN para possíveis eventos de *flash crowd*.

O gerenciamento da cache é composto de técnicas de *caching* usadas e frequência de atualização da cache, para a disponibilidade e confiabilidade do conteúdo.

3.2.3 O Redirecionador

O Redirecionador ou sistema de roteamento de requisições lidam com a tarefa de estimar qual o *surrogate* mais adequado para cada diferente requisição do cliente. O *surrogate* mais adequado pode nem sempre ser o próximo. Por consequência, o sistema de roteamento de requisições usa um conjunto de métricas tais como proximidade de rede, latência percebida pelo

cliente, distância, banda larga disponível, na tentativa de direcionar os usuários ao *surrogate* mais próximo que possa melhor servir sua requisição [21]. O redirecionador se comunica com o Gerenciador de Conteúdo para saber a localização das réplicas e informação de monitoramento coletadas através da CDN.

3.2.4 O Monitor

O papel do Monitor é verificar a rede para coletar dados que irão auxiliar as tarefas executadas pelo Redirecionador e pelo Gerenciador de Conteúdo. Isto inclui investigar o conjunto de servidores, saber a latência inerente à comunicação com os *surrogates* e servidor de origem, e adquirir informação sobre caminhos congestionados. Os monitores podem ser baseados em SNMP (*Simple Network Management Protocol*) ou baseados em agente. O monitor baseado em SNMP é visto como uma implementação direta. A desvantagem dessa técnica é ter pouca flexibilidade. Numa abordagem baseada em agente, a lógica é implementada no monitor e suas interações podem ser programadas e modificadas com o tempo. Além disso, os agentes podem ser organizados de forma hierárquica e alguns deles podem ser usados para agregar a informação reunida por outros. Além do mais, um conhecimento prévio sobre parte da topologia da rede poderia ser provido pelo operador e combinado com medidas ativas, para apresentar uma visão mais completa da topologia da rede e atuais condições.

3.2.5 O Gerenciador de Conteúdo

O Gerenciador de Conteúdo controla a forma com que os objetos são armazenados em cada *surrogate*. Ele provê essa informação ao Redirecionador, fazendo com que cada cliente seja servido pelo *surrogate* mais adequado. Além disso, a informação é gerenciada por um localizador de conteúdo e é armazenada no banco de dados de conteúdo. O localizador de conteúdo determina:

- O número de réplicas de um objeto;
- Em qual *surrogate* um novo objeto tem que ser armazenado;
- Política de eliminação de objetos não populares dos *surrogates*;
- Interação da CDN com os servidores de origem;
- Atualização dos objetos de multimídia nos *surrogates*;
- Quando e quais objetos de multimídia mover entre os surrogates.

3.2.6 Mecanismo de cobrança

Existem desafios técnicos e de negócio nos serviços de preço das CDNs. Provedores de CDN cobram seus clientes de acordo com o conteúdo entregue aos usuários finais pelos seus

servidores de réplica. O preço médio pelos serviços da CDN é relativamente alto [22], muitas vezes fora de alcance para empresas de nível médio ou organizações não-lucrativas. Os preços dos serviços da CDN variam de acordo com [9]:

- Uso de banda, que é medida pelo provedor de CDN para cobrar (por Mbps) os clientes;
- Variação da distribuição do tráfego, que caracteriza o preço sobre diferentes situações de congestionamento;
- Tamanho dos conteúdos replicados sobre os servidores, que é um critério crítico para emitir as cobranças;
- Número de servidores de borda, que captura a habilidade de uma CDN de oferecer conteúdo de forma que as taxas não excedam as cobranças em cenários típicos de *caching*;
- Questões de confiabilidade do sistema, estabilidade e segurança para encaminhamento dos conteúdos. Isto também inclui um custo de compartilhamento para dados confidenciais, que variam para cada CP, dado o tipo de proteção para estes dados.

As CDNs ajudam no mecanismo de contabilidade que coleta e localiza informações relatadas ao Redirecionador. Tal mecanismo reúne informação para cada componente da CDN. Esta informação pode ser usada nas CDNs para contabilidade, cobrança e propósitos de manutenção [23].

3.3 Categorias de CDN

3.3.1 CDN Convencional

As **CDNs tradicionais** foram as primeiras a serem comercializadas. Por conta da natureza proprietária existente para esse tipo de CDN, é difícil revelar informações em detalhe sobre as técnicas adotadas. É possível, no entanto, descrever algumas das principais características de algumas CDNs bem-sucedidas para fins de comparação. É observado que as mais bem-sucedidas CDNs são as da Akamai, Limelight Networks e Mirror Image.

A Akamai Techonologies foi fundada em 1998, em Massachusetts. O projeto envolvia uma pesquisa realizada no MIT com o objetivo de resolver o problema de *flash crowd* [24]. Quando seu serviço foi lançado, foi projetado para entregar apenas objetos de web (documentos e imagens). Desde então, ele envolvia entregar conteúdos dinâmicos (animações, scripts, DHTML) e *streaming* de áudio e vídeo.

3.3.2 CDN Acadêmica

Dentre as propostas de **CDNs acadêmicas**, existem a CoDeeN, Globule e Coral [25]. A CoDeeN [26] é uma CDN executada sobre a infraestrutura da PlanetLab. Esta CDN provê servidores proxy de alta performance, onde cada servidor desses se comporta tanto como um redirecionador de requisições como um servidor *surrogate*. Os nós da CoDeeN são desenvolvidos como proxies abertos com o fim de permitir acesso de fora da organização hospedeira.

A Globule [27] é uma CDN colaborativa de código aberto desenvolvida na Vrije Universiteit. Ela permite a operação da plataforma de hospedagem de provedores de conteúdo Web. Além disso, ela provê replicação, monitoramento de servidores e redirecionamento das requisições dos clientes aos *surrogates* disponíveis. Nela, um *site* é definido como uma coleção de documentos que pertencem a um usuário específico (ao proprietário) e um servidor é um processo executando em uma máquina conectada à rede, que executa uma instância do software da própria Globule. Cada *site* é composto de um servidor de origem, de backup, de réplica e de um redirecionador. A proximidade é medida com base na latência entre os nós.

A Coral [25] é uma CDN baseada em P2P, modelada para distribuir conteúdo Web. Esta CDN utiliza redirecionamento DNS para redirecionar as requisições próximas as webs caches. Estas *web caches* diminuem tanto a carga no servidor de origem como a latência percebida pelo usuário.

3.3.3 CDN Assistida por Ponto

A **CDN Assistida por Ponto** possui uma arquitetura híbrida que toma vantagem da CDN tradicional e de sistemas ponto-a-ponto. Uma CDN pode ser vista como uma arquitetura de entrega de conteúdo gerenciada centralmente, que se beneficia da administração profissional e recursos providos. Tais CDNs podem alcançar alta performance e confiabilidade, mas são bastante caras para escalar. Por outro lado, sistemas P2P devem contar com os recursos de seus "pontos", e as suas propriedades, por outro lado, exigem pouco investimento, onde na verdade, elas não requerem nenhuma infraestrutura, exibindo uma natureza descentralizada, mas pecando na qualidade de serviço (QoS) previsível.

Portanto, esta ideia é bastante atraente, combinando o melhor de cada mundo. Uma abordagem híbrida de um sistema P2P e CDN. Alguns autores propõem [28] uma arquitetura híbrida que integra CDN e uma distribuição multimídia de *streaming* baseada em P2P. As duas tecnologias de *streaming* se complementam: enquanto um arquivo precisa ser distribuído para um nicho de clientes, ele será antes distribuído pelo servidor CDN. À medida que atende as requisições dos clientes, o servidor da CDN também cria a semente, suprindo as partes em suas áreas de serviço. Combinados, o servidor CDN e os "pontos" (da arquitetura P2P) servem as requisições de *streaming* multimídia com uma capacidade maior que a de um único servidor.

3.3.4 CDN Telco

Provedores de conteúdo e ISP são entidades tradicionalmente independentes [29]: os ISPs proveem conectividade e são apenas responsáveis por transportar o conteúdo. No entanto, como na maioria dos sistemas de transporte, conectividade e largura de banda estão se tornando *commodities*, e os ISPs têm o seu lucro diminuindo. Eles querem participar no mercado de entrega de conteúdo e distribuir conteúdo aos seus clientes. A Internet, em perspectiva de negócios, tem se tornado mais centrada no conteúdo. Nesse contexto, muitas ISPs estão considerando desenvolverem suas próprias CDNs [30]. Além disso, desenvolver sua própria CDN permite ao ISP ter mais controle do tráfego em suas redes, diminuindo o consumo de banda larga e mantendo o tráfego localmente [31].

Dessa forma, é possível observar um novo tipo de CDN, desenvolvida pelo próprio ISP, capaz de prover espaço de armazenamento e com garantia de links aos seus clientes [32], formando então a **Telco-CDN**. Alguns exemplos dessa categoria de CDN são a Verizon e a Comcast.

3.3.5 CDN Federada

A disseminação dos provedores de CDN, principalmente ao considerar as oportunidades dadas às CDNs Telco, também traz a possibilidade de colaboração entre elas. Lazar e Terril [33] explicam alguns conceitos por trás das CDNs e como eles podem se conectar entre si de forma rentável. Entra então a ideia *CDN peering* [23], que é uma abordagem criada para reduzir os gastos e evitar impactos negativos no negócio, evitando a criação de "Ilhas de CDNs" que requerem um capital enorme. Cooperando entre si, provedores de CDN podem aumentar suas presenças geográficas, reduzir os custos e aumentar a disponibilidade, garantindo os acordos de serviço.

Uma federação de CDN é formada por um conjunto de CDNs autônomas que cooperaram entre si através de um mecanismo utilizado para compartilhar recursos enquanto tiram proveito de um maior alcance e uma maior escala. Este conjunto de CDNs autônomas pode ser chamada de **CDN Federada**.

3.3.6 CDN baseado em nuvem

Uma **CDN baseada em nuvem** pode reduzir o esforço e o custo de produzir uma solução para a entrega do conteúdo. Com a introdução da computação em nuvem, foi apresentado um crescimento das CDNs baseadas em nuvem. Ela se utiliza das nuvens para mapear elementos na infraestrutura da CDN em componentes virtuais. Por exemplo, um *surrogate* pode ser mapeado para um armazenamento IaaS (*Internet as a Service*) ou simplesmente para um serviço de entrega.

Na CDN baseada em nuvem, outro ator entra em cena: o provedor da infraestrutura da nuvem. O provedor da CDN ainda é o proprietário da infraestrutura, controlando os aspectos relativos à entrega do conteúdo, mas não no aspecto físico. O provedor da CDN, portanto, é agora não apenas responsável por garantir a entrega do conteúdo, mas também se torna responsável por alocar os recursos necessários na nuvem. Este modelo de CDN é bastante dinâmico.

3.3.7 CDN Autonômica

Uma **CDN Autonômica** toma vantagem de uma estratégia autonômica para gerenciamento dos *surrogates* e possui auto-organização baseada em funções. Um certo trabalho [34] propõe um algoritmo de alocação de réplicas distribuído, para alocação de réplicas de forma auto organizável. Cada *surrogate* executa o algoritmo de forma independente e encontra uma certa convergência para uma determinada alocação.

4 Fatiamento de Cache (Cache Slicing)

Esse capítulo trata da utilização do particionamento de armazenamento em um provedor CDN. É considerado um cenário onde múltiplos inquilinos (CPs) contratam uma CDN para encaminhar seus conteúdos. Nesse cenário, considera-se um processo de contrato entre o CP e a CDN na qual o primeiro contrata espaço de armazenamento para seus objetos, e o último se encarrega de distribuir este espaço entre seus *surrogates*, procurando fornecer a melhor distribuição possível. O conhecimento técnico desse processo de entrega de conteúdo permanece com a CDN. Ela é responsável por lidar com tarefas como gerenciamento de cache, redirecionamento de requisições, encaminhamento de conteúdo, bem como lidar com os requisitos de QoS. O provedor de conteúdo não pode determinar em qual cache o conteúdo será de fato armazenado. Ele apenas contrata um espaço de armazenamento e a CDN define como esse espaço será distribuído entre as caches.

Nesse capítulo apresentaremos o algoritmo de fatiamento de cache, originalmente proposto pelo trabalho proposto em [10]. Além disso, será apresentado um modelo de otimização com o fim de comparar seus resultados com os obtidos pela execução do algoritmo e suas variações.

4.1 O algoritmo de fatiamento de cache

Com base no modelo acima, o algoritmo de fatiamento de cache é apresentado. A CDN tem um número de *surrogates* (caches), que pode ser mudado, dada a decisão de se expandir sua infraestrutura. O processo de redução ou expansão da rede pode ser feito adquirindo um novo equipamento, por exemplo. Dada a alteração dessa infraestrutura, o algoritmo de fatiamento será executado mais uma vez.

O espaço de armazenamento para cada cache é dividido em porções pequenas de tamanho fixo, chamadas fatias (*slices*). As fatias têm tamanho fixo e este tamanho é um só para todas as CDNs e para todos os provedores de conteúdo (CP). Cada CP recebe uma porção de cada cache, com o fim de armazenar seus conteúdos. No entanto, ela não necessariamente possui uma porção em todas as caches. Sendo assim, cada CP recebe uma fatia de determinadas caches. É importante frisar que na abordagem que estamos utilizando é criada uma CDN virtual para o CP.

O número de fatias que um CP pode receber de cada cache depende da capacidade total que ele contrata. Portanto, em um cenário onde ele contrata 10 TB e a fatia tem tamanho 1 TB, este CP pode receber no máximo 10 fatias de todas as caches, fazendo possível uma configuração de disposição de fatias da seguinte forma: 2 fatias na cache #1, 4 fatias na cache

#2, 3 fatias na cache #3, 1 para a cache #4 e nenhuma para as caches restantes.

Sendo assim, o problema se trata de saber como distribuir as fatias em cada cache entre os CPs. O algoritmo proposto pode ser dividido em basicamente 3 etapas, que serão descritas a seguir.

4.1.1 Etapa 1

Nesta etapa, uma CDN virtual é formada definindo quantas fatias em quais caches cada CP receberá. Vale lembrar que esse número de fatias depende de quantas o provedor de conteúdo contratou inicialmente. Esta etapa pode ser considerada como uma etapa de disposição das fatias, devido ao fato de ser uma etapa similar à disposição geográfica das caches.

Para lidar com esse problema é necessário que a disposição das caches seja feita de forma ótima. Imaginemos o seguinte exemplo: um certo CP tem 20 fatias para distribuir nas caches que estão geograficamente distribuídas e sob controle do provedor de CDN. Esse provedor de CDN deve escolher onde tais fatias de cada CP devem ser dispostas, em cada um de suas caches disponíveis. É importante observar que uma vez que existem muitas fatias em cada cache, pode haver muitas fatias para um mesmo CP numa mesma cache física, o que é altamente desejável. É só imaginar que um CP tenha seus clientes concentrados numa região específica. Nessa situação, portanto, será apropriado que muitas fatias sejam alocadas numa cache localizada próxima a esses clientes.

Para fins de implementação, ordena-se os conteúdos, de um determinado CP, em um determinado *surrogate*, pela suas **popularidades**, fazendo com que seja priorizado o armazenamento de conteúdos mais populares, em uma certa região. Isso aumenta a taxa de acerto (*hitload*), aumentando por consequência o lucro obtido em se armazenar um determinado conteúdo em um determinado *surrogate*, e por consequência, a possibilidade do provedor desse conteúdo garantir uma fatia no *surrogate* em questão.

De forma resumida, nesta etapa, cada CP tem uma quantidade desejada de fatias para uma determinada cache. Entretanto, podem haver situações em que a soma de fatias desejadas por determinados CPs exceda a capacidade de uma determinada cache física. Neste caso, executa-se a Etapa 2 no algoritmo.

4.1.2 Etapa 2

A etapa 2 é executada apenas se, após a execução da etapa 1, o número de fatias alocadas para cada CP, para uma determinada cache, exceda a capacidade daquela cache. Dada a natureza elástica da infraestrutura, é bastante provável que essa etapa não aconteça. No entanto, caso ocorra, os CPs envolvidos nessa etapa terão de competir por disponibilidade, com o fim definir com quantas fatias naquela cache cada CP terá.

Para este trabalho, a estratégia de competição por fatias usada foi a Proporcional. Tal estratégia utiliza como critério de decisão **a popularidade de cada CP**. Portanto, o CP cujos objetos possuem mais requisições em uma determinada área deverá receber mais fatias na cache em questão. É importante notar que a popularidade de cada CP é considerada de forma independente, significando que cada gerenciador de CP irá definir a quantidade de fatias desejadas baseada apenas na popularidade de seus conteúdos.

Baseado no número de fatias desejadas por cada CP, é calculado o fator SP ou fator de Proporção de Fatia (*Slice Proportion*). Esse cálculo é dado pelo quociente entre o total de slots disponíveis na cache em questão e o somatório de fatias desejadas por todos os CPs.

$$SP = \frac{N° \, de \, slots \, disponíveis \, na \, cache}{\sum Fatias \, desejadas \, por \, todos \, os \, CPs} \tag{4.1}$$

Calculado o fator SP, multiplicamos ele pelo número de fatias desejadas por cada CP, obtendo por fim a quantidade de fatias definitiva que cada CP receberá naquela cache. Um exemplo pode ser observado na imagem abaixo (Figura 4) [10].

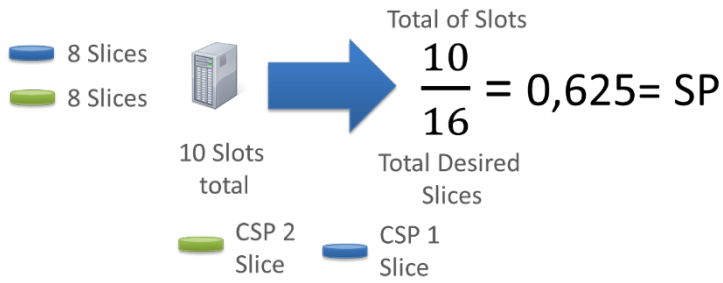


Figura 4 – Competição por fatias em uma determinada cache

Nesse exemplo, cada CP ficará com 5 fatias na cache que está em competição. Por conta desse processo de competição, após a execução da etapa 2, pode haver algum CP que pediu por uma fatia numa cache e não conseguiu. Para lidar com as fatias não alocadas, a etapa 3 é executada, com o fim de manipular cada uma dessas fatias.

4.1.3 Etapa 3

A etapa 3 é uma fase simples, onde usamos basicamente duas estratégias possíveis para lidar com ela. A primeira estratégia é a execução de um algoritmo *Round Robin*, que distribui as fatias restantes entre todas as caches e, a cada rodada, aloca uma fatia por cache. A segunda estratégia é essencialmente igual à primeira. No entanto, ela tem uma pequena modificação: os custos de carga para cada CP em cada cache são calculados e ordenados. Logo após, as fatias são distribuídas, uma por vez, dado o rank dos cálculos acima, privilegiando as caches que possuem mais requisições.

É importante destacar que, após a execução dessas três etapas, assume-se a CDN realizou a distribuição da forma mais adequada possível, levando sempre em conta a popularidade dos conteúdos em determinadas regiões.

4.2 Modelo de Otimização

Para a modelagem desse problema, teremos um conjunto K de provedores de conteúdo e um conjunto J de *surrogates*. Para cada provedor de conteúdo $k \in K$ e cada *surrogate* $j \in J$, são apresentadas as variáveis contínuas não negativas s_{k_j} e l_{k_j}. s_{k_j} descreve o número de fatias do provedor de conteúdo k na cache j. l_{k_j} descreve o load do provedor de conteúdo k na cache j. Para descrever o número de fatias contratadas por cada provedor de conteúdo k nós utilizamos N_k. Para descrever a capacidade da cache j nós utilizamos C_j.

O número de fatias alocadas deve ser igual ao número de fatias contratadas pelo provedores de conteúdo.

$$\sum_{j} s_{k_j} = N_k \quad \forall k \tag{4.2}$$

O número de fatias na cache deve ser menor que a sua capacidade de armazenamento.

$$\sum_{j} s_{k_j} \leq C_j \quad \forall j \tag{4.3}$$

O domínio da variável de decisão deve estar nos naturais.

$$s_{k_j} \in \mathbb{N} \tag{4.4}$$

O objetivo é, portanto, maximizar o lucro que se obtém com a disposição das fatias, com a intenção de atrair as fatias para as caches de maior demanda. Isto pode ser apresentado como

$$\sum_{k} \sum_{j} \quad l_{k_j} * s_{k_j} \quad \forall k \; \forall j \tag{4.5}$$

4.3 Ambiente de execução dos algoritmos

Para execução dos algoritmos, foi desenvolvida uma espécie de simulador, na linguagem **C**, onde foram definidos os seguintes elementos:

- Estrutura de dados que representam os componentes e atores da CDN;
- Geração das requisições por parte dos clientes aos conteúdos de cada CP, feita através do povoamento de uma matriz binária de requisições;
- Geração dos conteúdos (tamanho e popularidade);

- Métodos para cada um dos algoritmos implementados.

Os dados relativos a arquitetura da CDN (tais como coordenadas geográficas dos elementos, capacidade contratada pelo CP, capacidade do *surrogate*, etc), bem como os valores de *load* para cada CP em cada *surrogate*, são exportados e utilizados pelo *solver*, a partir de um arquivo em Python, gerado na própria simulação.

Para os cálculos de *hitload* relativos ao experimento do modelo de otimização, obtêm-se a configuração das fatias distribuídas entre os *surrogates* (saída gerada pelo *solver*), onde essas informações são importadas pelo simulador que efetua, por fim, os cálculos necessários.

5 Implementação e Resultados

Neste capítulo serão apresentados detalhes sobre as implementações realizadas bem como os resultados obtidos ao se executar um conjunto de experimentos. Tais experimentos serão discutidos logo em seguida.

5.1 Implementação

Nesta etapa, foi implementado o algoritmo apresentado no capítulo anterior. Para fins de obtenção dos resultados foram realizadas as seguintes modificações:

- Alocação aleatória das fatias: o tamanho das fatias é calculado normalmente, assim como para os outros algoritmos. No entanto, as fatias contratadas pelos provedores de conteúdo são dispostas de forma aleatória dentre os *surrogates*
- Alocação utilizando o algoritmo de fatiamento de cache: neste experimento, é executado o algoritmo apresentado no capítulo anterior. Na terceira etapa desse algoritmo, é utilizado o algoritmo *Round Robin* para distribuição do restante das fatias relativas a essa etapa.
- Alocação utilizando o algoritmo de fatiamento de cache adaptado: executa-se o algorimo apresentado no capítulo anteior, com a terceira etapa modificada. Ao invés de ser utilizar o algoritmo *Round Robin*, os custos de carga para cada CP em cada *surrogate* são calculados e a partir disso, são ordenados. Sendo assim, a alocação do restante das fatias realiza-se privilegiando os caches que possuem mais requisições para objetos daquele CP.

Além das implementações dos algoritmos acima, foi realizada a implementação do modelo de otimização apresentado no capítulo anteior. Para resolver o problema modelado, foi utilizado o **Gurobi**[1], um *solver* de otimização para programação matemática. O Gurobi foi modelado para atender diversas arquiteturas, usando as mais avançadas implementações dos algoritmos mais atuais. Problemas de programação linear e programação quadrática são exemplos de problemas resolvidos por esse *solver* (que na verdade possui um conjunto de *solvers* em sua arquitetura).

Como resultado dessa modelagem, o Gurobi nos apresenta uma determinada configuração de alocação de fatias de cada CP para cada *surrogate*, a partir da maximização da função de lucro (Equação 5.5). A partir disso, calculamos a taxa de *hitload* com base nessa configuração, levando em conta o *load* em cada um desses *surrogates*.

[1] http://www.gurobi.com

Neste trabalho, executaremos alguns experimentos variando os seguintes parâmetros:

- Quantidade de provedores de conteúdo (CPs);
- Quantidade de *surrogates* espalhados geograficamente;
- Quantidade de clientes que fazem uma série de requisições.

Na tabela abaixo, podemos observar alguns dos valores que foram utilizados para os experimentos:

Tabela 1 – Tabela de valores para os experimentos

# de Surrogates	**# de CPs**	**# de Clientes**
(5, 10, 20)	(2, 4, 8, 16, 24, 32, 36, 40, 50)	(500,1K, 2K)

Além desses, alguns valores foram mantidos constantes. Alguns desses valores são:

- Capacidade contratada por cada CP: **5 GB**
- Capacidade de cada *surrogate*: **10 GB**
- Tamanho da fatia (slice): **1 GB**

5.2 Resultados

Nessa seção, apresentaremos os resultados obtidos a partir da execução de cada um dos algoritmos citados no início desse capítulo. Além disso, logo após apresentarmos os resultados obtidos pela implementação do modelo de otimização, faremos uma comparação entre os valores obtidos através do modelo com os valores obtidos através dos algoritmos. A métrica utilizada para comparação foi a razão **Q**, apresentada abaixo:

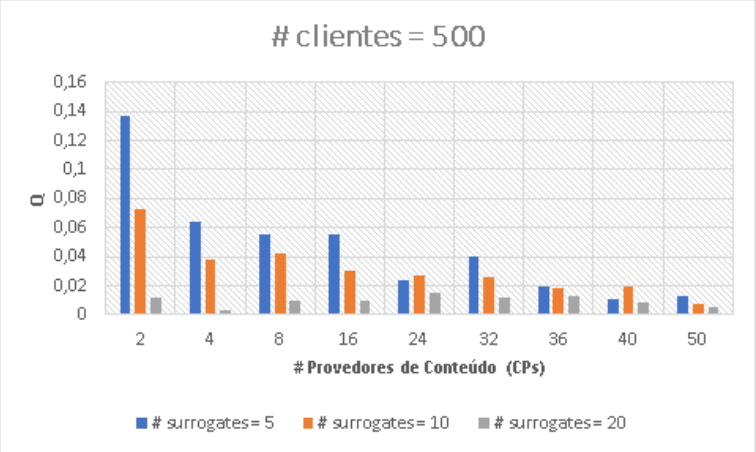
$$Q = \left(\frac{1}{N_K}\right) \sum_{k} \left(\left(\frac{1}{N_J}\right) \sum_{j} \frac{h_{k_j}}{l_{k_j}} \right) \tag{5.1}$$

N_K e N_J são respectivamente o número de provedores de conteúdo (CPs) e o número de *surrogates*. h_{k_j} é taxa de *hitLoad* no *surrogate* j para o CP k. A variável l_{k_j} (taxa de *load*) já foi apresentada no capítulo anterior.

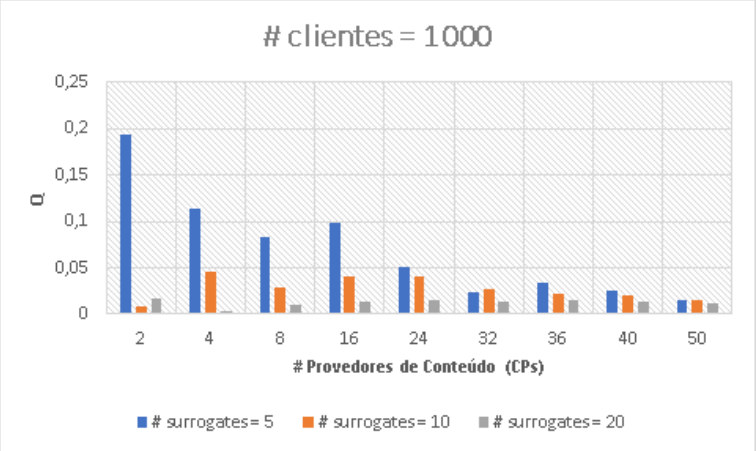
Os valores de *Q* obtidos através das execuções dos algoritmos serão posteriormente comparados com os valores obtidos através do modelo de otimização. Esses valores obtidos através do modelo de otimização serão utilizados como valores "ideais" para comparações posteriores.

Seguem os gráficos obtidos pela execução de cada um dos algoritmos e do modelo de otimização.

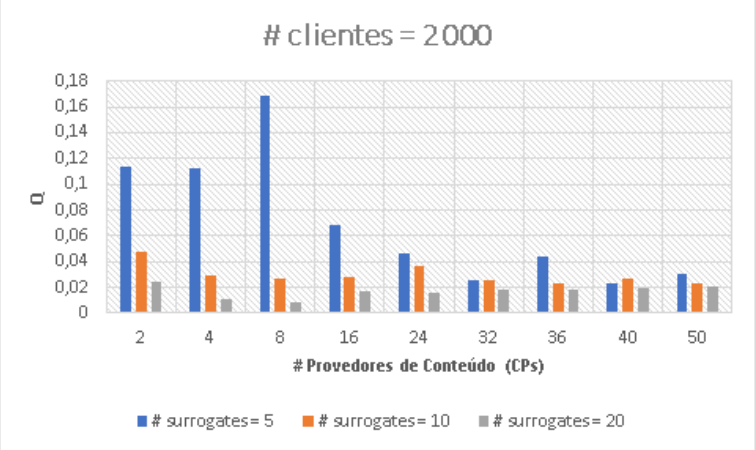
- **Experimento 1** : Algoritmo com alocação aleatória das fatias



(a) # Clientes = 500



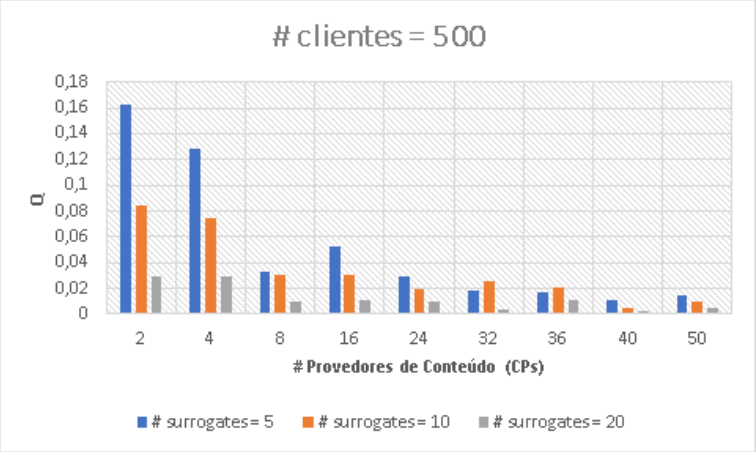
(b) # Clientes = 1000



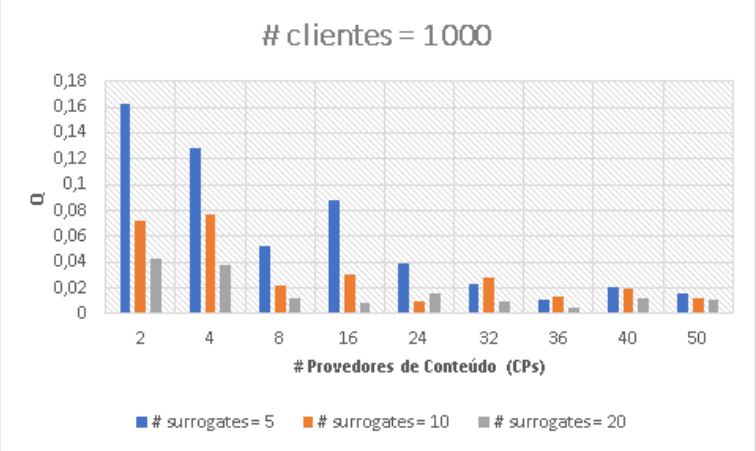
(c) # Clientes = 2000

Figura 5 – Experimento 1 : (Valor **Q**) x (# Provedores de Conteúdo)

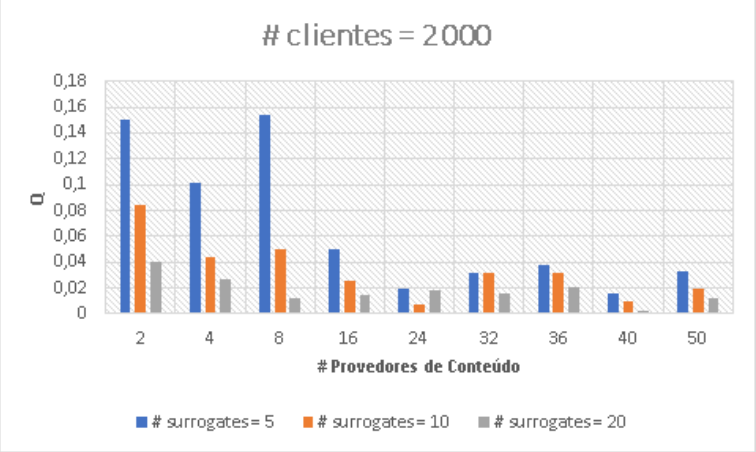
- **Experimento 2** : Algoritmo com alocação utilizando o algoritmo de fatiamento de cache



(a) # Clientes = 500



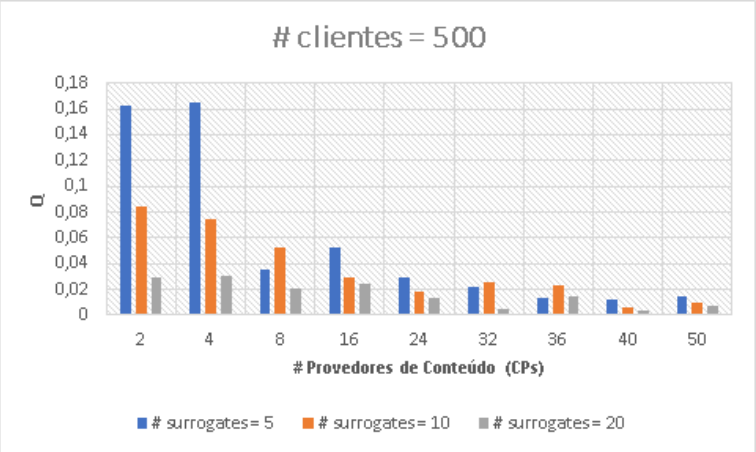
(b) # Clientes = 1000



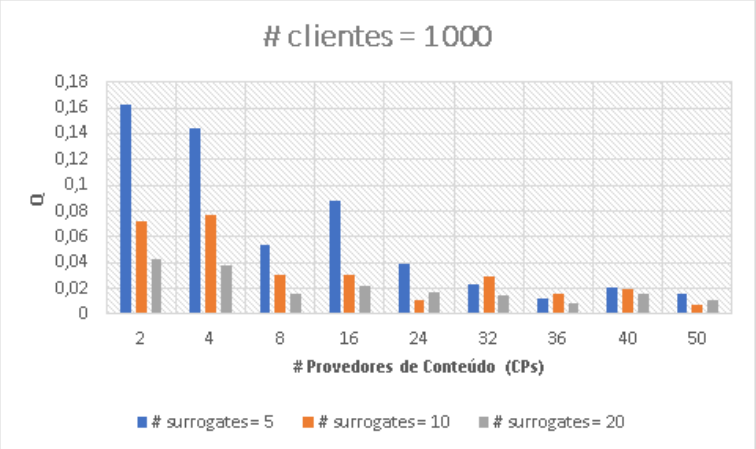
(c) # Clientes = 2000

Figura 6 – Experimento 2 : (Valor **Q**) x (# Provedores de Conteúdo)

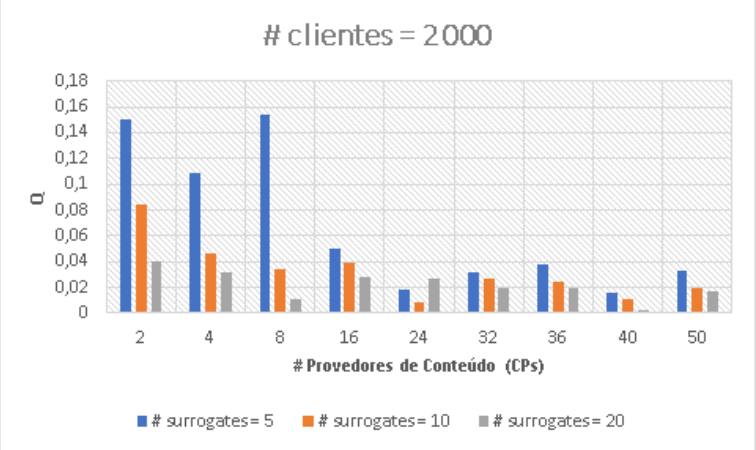
- **Experimento 3** : Algoritmo com alocação utilizando o algoritmo de fatiamento de cache adaptado



(a) # Clientes = 500



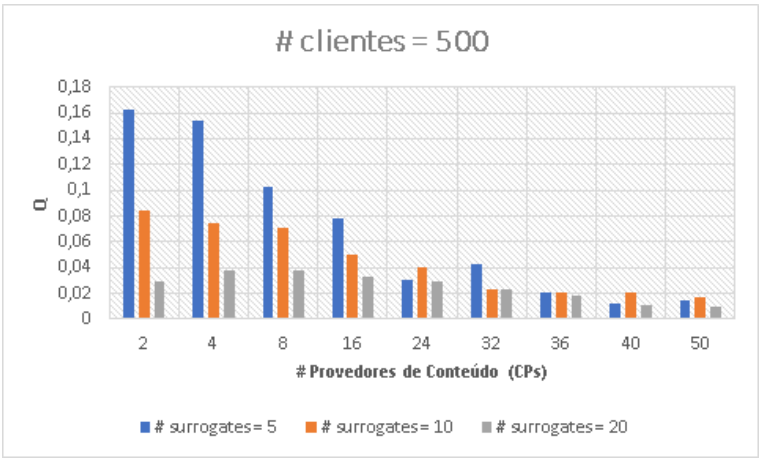
(b) # Clientes = 1000



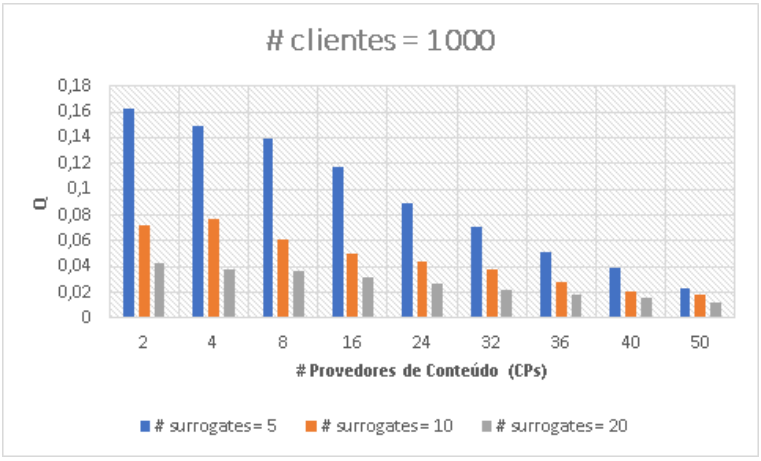
(c) # Clientes = 2000

Figura 7 – Experimento 3 : (Valor **Q**) x (# Provedores de Conteúdo)

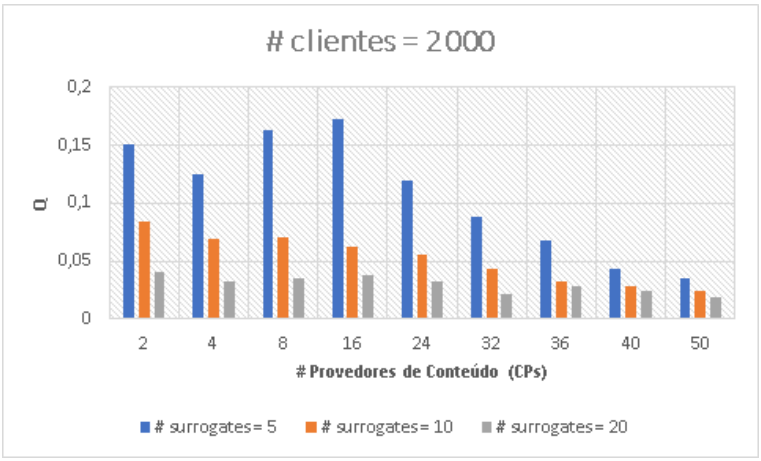
- **Experimento 4** : Resultados obtidos a partir do modelo de otimização



(a) # Clientes = 500



(b) # Clientes = 1000



(c) # Clientes = 2000

Figura 8 – Experimento 4 : (Valor **Q**) x (# Provedores de Conteúdo)

Com base nos resultados apresentados acima, podemos então fazer uma comparação dos resultados obtidos nos três primeiros experimentos com os resultados extraídos do *solver* (Experimento 4). Na tabela abaixo, verificamos a média dos **erros absolutos** para cada um dos experimentos. Os erros absolutos calculados apresentam a diferença aritmética entre os valores obtidos através do modelo de otimização (valor ideal) e os valores obtidos através execuções dos algoritmos.

Tabela 2 – Erro absoluto entre os valores obtidos através do modelo de otimização (valor ideal) e os valores obtidos pelas execuções dos algoritmos

# de Clientes	**I. Alocação aleatória**	**II. Cache Slicing Puro**	**III. Cache Slicing Modificado**
500	0,01697	0,01383	**0,01040**
1000	0,01863	0,01915	**0,01702**
2000	0,02421	0,02394	**0,02287**
Média	0,01994	0,01897	0,01676
Desvio Padrão	0,00301	0,00413	0,00624

6 Conclusão

Este trabalho teve como objetivo apresentar uma estratégia para otimização da alocação de armazenamento para provedores Redes de Distribuição de Conteúdo. Técnicas como as que foram apresentadas nesse trabalho vêm sendo utilizadas por várias instituições ao redor do mundo, sendo continuamente alvo de estudos para avaliação e aplicação de melhorias. A utilização de um meio autônomo para alocação de armazenamento em caches geograficamente distribuídas mostrou ser bastante eficiente, principalmente em cenários onde o número de clientes, componentes de armazenamento e provedores de conteúdo são relativamente altos.

Através da técnica proposta, foram feitas variações no algoritmo de *cache slicing* original, sendo possível extrair bons resultados, dado os diferentes cenários utilizados como entrada pelo algoritmo. Além disso, a concepção e utilização de um modelo de otimização nos permitiu fazer uma melhor análise do problema, sendo por fim utilizada como padrão comparativo, verificando (através do erro relativo) o quão distantes os resultados obtidos através dos algoritmos estavam do resultado ótimo, encontrado através do modelo de otimização utilizado.

Dentre as principais dificuldades encontradas nesse projeto, podemos citar: (i) a concepção da ideia de como as fatias de cache poderiam ser melhor distribuídas entre as *caches*, para cada provedor de conteúdo; (ii) a definição de um modelo de otimização que caracterizasse de forma correta o problema formulado; (iii) o *solver* não conseguia lidar com altos valores de *load* para serem utilizados nos cálculos, ocasionando a partir de um certo momento na sobrecarga do *solver*, dado um cenário com um número elevado de CPs e *surrogates*.

Além de realizar melhorias nos algoritmos implementados nesse trabalho, pretende-se estudar mais técnicas que possam apresentar melhores resultados, bem como aplicar melhorias no modelo de otimização utilizado na modelagem desse problema.

Referências

[1] I. Gartner, "Gartner says 8.4 billion connected "things"will be in use in 2017, up 31 percent from 2016." http://www.gartner.com/newsroom/id/3598917, 2017. Accessed: 2017-06-08. Citado na página 17.

[2] R. K. S. Erik Nygren and J. Sun, "The akamai network: A platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, pp. Pages 2–19, July 2010. Citado na página 17.

[3] M. I. Internet, *Content Delivery and the Mirror Image Adaptive CAP Network, 2007*. www.mirror-image.com, 2007. Citado na página 17.

[4] F. H. Dudouet, P., S. Ruiz, A. Gomes, and T. Bohnert, "A case for cdn-as-a-service in the cloud: A mobile cloud networking argument," in *International Conference on Advances in Computing, Communications and Informatics*, pp. 651–657, ICACCI 2014, September. Citado na página 18.

[5] R. Buyya, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, pp. 599–616, 2003. Citado na página 18.

[6] F. Chen, K. Guo, J. Lin, and T. La Porta, "Intra-cloud lightning: Building cdns in the cloud," in *2012 Proceedings IEEE*, vol. 433, pp. 651–657, INFOCOM, March 2012. Citado na página 18.

[7] T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, and G. Urvoy-Keller, "From content distribution networks to content networks - issues and challenges," *Computer Communications*, vol. 29, 2003. Citado na página 21.

[8] S. Das and J. Kangasharju, "Evaluation of network impact of content distribution mechanisms," in *InfoScale '06 Proceedings of the 1st international conference on Scalable information systems*, no. 35, 2006. Citado na página 21.

[9] G. Pallis and Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, p. 101–106, 2006. Citado 3 vezes nas páginas 23, 29 e 31.

[10] A. L. C. Moreira, *An Adaptable Storage Slicing Algorithm for Content Delivery Networks*. PhD thesis, UFPE, 2015. Citado 3 vezes nas páginas 23, 35 e 37.

[11] G. Peng, "Cdn: Content distribution network," technical report tr-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, 2003. Citado na página 26.

[12] A. K. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," tech. rep., Univ. of Melbourne, 2007. Citado na página 27.

[13] J. F. Barnes and R. Pandey, "Cachel: Language support for customizable caching policies," in *Proc. Fourth Int'l Web Caching Workshop (WCW)*, 1999. Citado na página 27.

[14] B. Wu and A. Kshemkalyani, "Objective-optimal algorithms for long-term web prefetching," *IEEE Trans. Computers*, vol. 55, no. 1, pp. 2–17, 2006. Citado na página 27.

[15] Y. Chen, L. Qiu, W. Chen, and L. Nguyen, "Efficient and adaptive web replication using content clustering," *IEEE J. Selected Areas in Comm.*, vol. 21, no. 6, pp. 979–994, 2003. Citado na página 27.

[16] S. Jamin, C. Jin, Y. Jin, D. Raz, S. Y., and L. Zhang, "On the placement of internet instrumentation," *InProc. of IEEE INFOCOM*, p. 295–304, 2000. Citado na página 28.

[17] Y. Bartal, "Probabilistic approximation of metric space and its algorithmic applications," in *In Proceedings of 37th Annual IEEE Symposium on Foundations of Computer Science*, October 1999. Citado na página 28.

[18] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transaction on Networking*, vol. 8, no. 5, 2000. Citado na página 28.

[19] S. Jamin, C. Jin, A. R. Kure, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *In Proceedings of IEEE INFOCOM*, April 2001. Citado na página 28.

[20] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," *In Proceedings of IEEE INFOCOM*, pp. 1587–1596, April 2001. Citado na página 29.

[21] K. Delgadillo, "Cisco distributed director," tech. rep., Cisco Systems, June 1997. Citado na página 30.

[22] D. Rayburn, "Cdn research data: Market sizing and pricing trends," *Streaming Media West: The Business and Technology of Online Video*, 2009. Citado na página 31.

[23] R. Buyya, A. K. Pathan, J. Broberg, and Z. Tari, "A case for peering of content delivery networks," *IEEE Distributed Systems Online*, p. 9, 2006. Citado 2 vezes nas páginas 31 e 33.

[24] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Computing*, p. 50–58, 2002. Citado na página 31.

[25] "Windows NT coralcdn." http://www.coralcdn.org/. Accessed: 2017-06-02. Citado na página 32.

[26] L. Wang, K. S. Park, R. Pang, V. S. Pai, and L. Peterson, "Reliability and security in codeen content distribution network," in *In Proceedings of the USENIX 2004 Annual Technical Conference*, June 2004. Citado na página 32.

[27] Pierre and M. van Steen, "Globule: A collaborative content delivery network," *IEEE Communications*, vol. 44, August 2006. Citado na página 32.

[28] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a cdn-p2p hybrid architecture for cost-effective streaming media distribution," *Multimedia Systems*, vol. 11, pp. 383–399, April 2006. Citado na página 32.

[29] K. Delgadillo, "Cooperative content distribution and traffic engineering in an isp network," tech. rep., Princeton University, 2009. Citado na página 33.

[30] A. Sharma, A. Venkataramani, and R. Sitaraman, "Distributing content cimplifies isp traffic engineering." http://arxiv.org/abs/1209.5715, 2012. Accessed: 2017-05-25. Citado na página 33.

[31] Z. Li, M. K. Sbai, Y. Hadjadj-Aoul, A. Gravey, D. Alliez, J. Garnier, G. Madec, G. Simon, and K. Singh, "Network friendly video distribution," in *International Conference on the Network of the Future*, vol. 13, 2012. Citado na página 33.

[32] D. Rayburn, "More isps not letting cdn place servers inside their network, doing it themselves." http://blog.streamingmedia.com/the_business_of_online_vi/2009/04/more-isps-not-letting-cdn-place-servers-inside-their-network-doing-it-themselv
html, 2009. Citado na página 33.

[33] I. Lazar and W. Terrill, "Exploring content delivery networking," *IEEE IT Professional*, vol. 3, August 2001. Citado na página 33.

[34] K. Herrmann, "Self-organizing replica placement - a case study on emergence," in *Self-Adaptive and Self-Organizing Systems - SASO*, pp. 9–11, July 2007. Citado na página 34.