Uma Heurística Distribuída para o Problema de Atribuição de Clientes a Servidores em Redes de Distribuição de Conteúdos

Rafaelli de Carvalho Coutinho, Lúcia M. A. Drummond,
Yuri Frota, Luidi Simonetti
Instituto de Computação – Universidade Federal Fluminense
Rua Passo da Pátria 156 - Bloco E - 3º andar - São Domingos - 24210-240 - Niterói - RJ
{rcoutinho,lucia,yuri,luidi}@ic.uff.br

RESUMO

Uma Rede de Distribuição de Conteúdo (RDC) é uma rede sobreposta que mantém réplicas de conteúdos em servidores com o objetivo de diminuir o atraso, a carga dos servidores e o congestionamento da rede, melhorando a Qualidade de Serviço percebida pelos clientes. Neste trabalho, uma heurística distribuída é proposta para o Problema de Atribuição de Clientes a servidores (PAC), onde dadas as requisições dos clientes por conteúdos, o objetivo é fornecer as atribuições cliente-servidor de maneira a reduzir o custo de comunicação total entre eles. Uma formulação matemática do problema baseada no Problema de Transporte é descrita. O algoritmo proposto, chamado DistPAC, é baseado nas heurísticas Canto Noroeste e Custo Mínimo, que são tradicionalmente usadas para a obtenção de uma solução inicial viável para o Simplex de Transporte. Os experimentos mostraram que o DistPAC obtém soluções próximas às ótimas em tempos baixos, trocando um número razoavelmente pequeno de mensagens.

PALAVRAS CHAVE. Redes de Distribuição de Conteúdo, Problema de Transporte, Algoritmos Distribuídos.

Área Principal: PO em Telecomunicações e Sistemas de Informações

ABSTRACT

A Content Distribution Network (CDN) is an overlay network that replicates contents in servers with the objective of reducing the delay, server load and network congestion, improving the Quality of Service perceived by clients. In this paper, a distributed heuristic is proposed for the Problem of Assigning Clients to Servers (PAC), where, given the requests of clients for contents, the goal is to provide client-server assignments to reduce the total cost of communication between them. A mathematical formulation of the problem based on the Transportation Problem is described. The proposed algorithm, called DistPAC, is based on Northwest Corner and Minimum Cost heuristics, which are traditionally used to obtain an initial feasible solution for the Transportation Simplex. Experiments showed that the DistPAC obtains solutions close to the optimum in short times, with a reasonable low number of exchanged messages.

KEY WORDS. Content Distribution Network, Transportation Problem, Distributed Algorithms.

Main area: OR on Telecommunications and Information Systems

1 Introdução

Com a popularização da internet, o número de usuários aumentou bastante nos últimos anos, e consequentemente, a demanda por conteúdos na rede. Assim, pesquisas têm sido realizadas para conseguir atender, com Qualidade de Serviço (QoS), o crescente número de requisições por conteúdos. As pesquisas se concentram na utilização de Redes de Distribuição de Conteúdos (*Content Distributed Network*) para o fornecimento dos conteúdos aos usuários finais (clientes). Uma Rede de Distribuição de Conteúdo (RDC) é um sistema de computadores que possui cópias de conteúdos posicionadas em vários pontos de uma rede com o objetivo de maximizar a utilização da largura de banda (i.e. o acesso a estes dados) por clientes em toda a rede. Dentre os diversos tipos de conteúdos pode-se citar: objetos da Web, objetos disponíveis para download (arquivos de mídia, software, documentos) e aplicações *streaming* de mídia em tempo real.

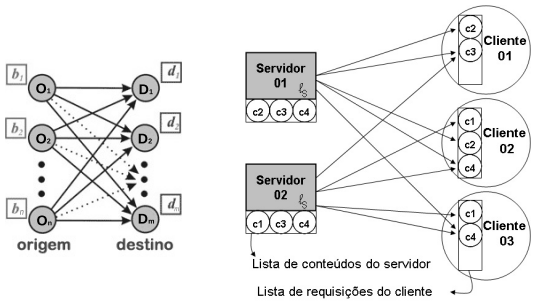
Com o objetivo de implementar o serviço de uma RDC, uma série de problemas precisam ser considerados. Um deles é o de encontrar as melhores localizações em uma rede para posicionar os servidores. Este problema é denominado Problema de Posicionamento de Servidor (PPS) (Bektas *et al.*, 2007). Um outro problema inerente a RDC é o Problema de Posicionamento de Réplicas (PPR) que consiste em encontrar os melhores servidores dentro da rede para colocar as réplicas dos conteúdos, e desta forma, atender as requisições (Zhou e Xu, 2007). Para isto, é necessário o conhecimento das posições dos servidores, o tamanho dos conteúdos e as informações sobre as requisições. O Problema de Atribuir Clientes a servidores (PAC) consiste em, dadas as requisições dos clientes e os posicionamentos das réplicas dos conteúdos, encontrar as atribuições cliente-servidor (i.e. que servidor irá atender que cliente) considerando a disponibilidade dos conteúdos, a carga nos servidores e o custo de comunicação dos enlaces.

Neste artigo, o PAC foi estudado e uma nova modelagem foi proposta baseada no Problema de Transporte (Ahuja *et al.*, 1993). Este problema é muito estudado na área de fluxo de redes e pode ser resolvido de forma exata por programação linear através de um algoritmo especializado conhecido como Simplex de Transporte (Dantzig, 1951). O Simplex de Transporte requer que uma solução inicial viável, de preferência de boa qualidade, seja calculada. Tipicamente, os métodos do Canto Noroeste ou do Custo Mínimo são utilizados para encontrar esta solução inicial. Note que em uma RDC as requisições estão distribuídas entre os servidores e podem sofrer modificações no decorrer do tempo. Assim, o custo para coletar, armazenar e atualizar estas informações em um nó central pode ser proibitivo. Neste contexto, a utilização de algoritmos distribuídos executados em servidores que possuem um conhecimento limitado sobre a RDC pode ser uma alternativa atraente para a solução do PAC com eficiência. As principais contribuições desse trabalho se resumem nos seguintes pontos: (i) modelagem do PAC como um Problema de Transporte; (ii) adaptação dos algoritmos sequenciais do Canto Noroeste e do Custo Mínimo para solução do PAC; e (iii) proposta da heurística distribuída, DistPAC, e avaliação experimental sobre um conjunto de instâncias da literatura.

2 Trabalhos Relacionados

Bektas et al. propuseram abordagens heurísticas e exatas para os problemas PPS, PPR e PAC com o objetivo de minimizar o custo total de distribuição (Bektas *et al.*, 2007).

A seguir, em Bektas *et al.* (2008), duas novas abordagens são apresentadas: uma baseada em decomposição de Benders e outra baseada em relaxação Lagrangeana e decomposição, para a solução do problema. Em Mundur e Arankalle (2006) são apresentados uma formulação matemática baseada no problema de transporte para o problema PPS, e dois novos protocolos de redes para o PAC. Shah et al. também abordam o PAC, atribuindo clientes a servidores de forma a minimizar os atrasos de comunicação (Shah *et al.*, 2005). Foram propostos dois algoritmos. O primeiro é baseado em um algoritmo para o problema de *matching*, utilizando fluxo máximo de



(a) (b)

Figura 1: (a) Grafo bipartido completo (b) Atribuições cliente-servidor

custo mínimo e o segundo, em um algoritmo para o problema de *stable-marriages*. Em Neves *et al.* (2010), é apresentada uma formulação matemática para a versão *off-line* do PPR e PAC, assim como uma heurística híbrida para a versão *on-line* destes problemas.

Além destes trabalhos relacionados a RDC, um estudo sobre o método Simplex foi realizado com o objetivo de compreender as abordagens distribuídas e paralelas existentes (Hall, 2010). Entretanto os trabalhos relacionados encontrados resolvem o PAC juntamente com o problema de posicionamento de servidores e de seleção, através de métodos matemáticos ou por heurísticas sequenciais. Não foram feitas comparações com estes trabalhos, pois pode não ser justo já que o posicionamento de servidores foi obtido de forma ótima para o PAC. Além disso, não foram encontradas abordagens paralelas ou distribuídas para o Simplex de Transporte. Assim, a contribuição deste trabalho consiste em apresentar uma heurística distribuída baseada nos métodos de obtenção de uma solução inicial viável para o Simplex de Transporte para resolver o PAC. No entanto, esta heurística também constitui uma primeira etapa para a solução do Simplex de Transporte.

3 Definição do Problema

O PAC consiste em atribuir as requisições de clientes aos servidores de maneira que todas as requisições sejam atendidas. Algumas considerações feitas sobre a RDC: a rede possui conteúdos pré-posicionados; podem existir cópias de conteúdos em servidores distintos e os servidores possuem largura de banda limitada. Neste trabalho, o modelo matemático foi baseado no Problema de Transporte, que é um problema de otimização, em que itens devem ser transportados de pontos de origem para pontos de destino, as origens possuem ofertas limitadas do item e os destinos, demandas. O objetivo é minimizar o custo de transporte. Este problema pode ser definido sobre um grafo bipartido completo $G = (V, A)$, em que o conjunto de vértices V está associado aos n pontos de origem e m pontos de destino, e o conjunto de arcos A possui um arco para cada par origem-destino. Assim, para cada vértice origem i, b_i indica seu limite de oferta, para cada vértice destino j, d_j é a sua demanda e para cada arco entre a origem i e o destino j existe um custo c_{ij} associado. A Figura 1(a) ilustra este grafo. Este problema pode ser utilizado para modelar o PAC da seguinte maneira. Os servidores constituem os vértices de origem e as requisições dos clientes, os vértices de destino. As demandas e as ofertas são, respectivamente, a largura de banda do conteúdo (requisito de QoS do conteúdo) e a largura de banda do servidor. Por exemplo, na Figura 1(b), existem 2 servidores. Um servidor possui cópia dos conteúdos 2, 3 e 4 e o outro possui cópia dos conteúdos 1, 3 e 4. Existem, também, 7 requisições por conteúdos de 3 clientes distintos. O cliente 1 possui requisições dos conteúdos 2 e 3, o cliente 2, requisições dos conteúdos 1, 2 e 4 e o cliente 3, requisições dos conteúdos 1 e 4. No entanto, os vértices e os arcos que formam o PAC (Figura 1(b))

não representam um grafo bipartido completo como no Problema de Transporte, pois nem todos os servidores possuem cópias de todos os conteúdos. Assim, para modelar o PAC baseando-se no Problema de Transporte foi necessário inserir arcos artificiais com custos de comunicação infinito entre vértices origem e destino não adjacentes. Estes custos infinitos indicam que o conteúdo da requisição não está presente no servidor associado. Os demais arcos têm o custo correspondente ao atraso do enlace. Assim, o objetivo é atribuir clientes a servidores de forma a minimizar o custo de comunicação.

3.1 Formulação Matemática

Nesta seção, a formulação para o PAC como um Problema de Transporte é apresentada formalmente. A notação abaixo foi utilizada na formulação do problema:

Conjuntos	
N	Conjunto de servidores, indexado por i
M	Lista de requisições, indexada por r
K	Conjunto de clientes, indexado por j
C	Lista de conteúdos, indexada por c
R_j	Lista de requisições do cliente j pertencente a M
C_i	Lista de conteúdos do servidor i
Dados	
$c(r)$	Conteúdo da requisição r
$j(r)$	Cliente da requisição r
d_r	Banda do conteúdo da requisição r
b_i	Largura de banda disponível no servidor i
c_{ij}	Custo de comunicação do enlace (i, j)
Variável	
$x^r_{ij(r)}$	Tráfego da requisição r que passa pelo enlace $(i, j(r))$

O custo de comunicação entre servidores i e j é infinito quando o conteúdo $c(r)$ da requisição r de j não pertence a C_i. O objetivo do problema é determinar o tráfego de cada requisição que passa por cada enlace minimizando o custo de comunicação (Equação 1). Para isso, o tráfego total de uma requisição por um conteúdo tem que ser equivalente a largura de banda deste conteúdo (Equação 2) e o tráfego de todas as requisições atendidas por um servidor não pode exceder a sua banda (Equação 3). Assim, a formulação é escrita da seguinte maneira:

$$min \sum_{i \in N} \sum_{j \in K} \sum_{r \in R_i} c_{ij} x^r_{ij} \tag{1}$$

sujeito a:

$$\sum_{i \in N} x^r_{ij} = d_r, \forall j \in K, r \in R_j \tag{2}$$

$$\sum_{j \in K} \sum_{r \in R_j} x^r_{ij} \leq b_i, \forall i \in N \tag{3}$$

$$x^r_{ij} \geq 0, \forall i \in N, j \in K, r \in R_j \tag{4}$$

4 Algoritmo Sequencial

O Problema de Transporte pode ser representado por uma tabela, como por exemplo a Tabela 1. Nesta tabela, as linhas representam os servidores e as colunas, as requisições dos clientes

por um conteúdo específico. As células são formadas pelo custo de transporte associado aos índices linha e coluna, ou seja, o custo de transporte do vértice linha para o vértice coluna. Cada célula também possui uma variável $x^r_{ij(r)}$ associada ao tráfego da requisição r entre o servidor i e o cliente $j(r)$. O método do Canto Noroeste para encontrar uma solução inicial viável para o Simplex de

	D_1	D_2	...	D_m	**ofertas**
O_1	c_{11}	c_{12}	...	c_{1m}	b_1
O_2	c_{21}	c_{22}	...	c_{2m}	b_2
$\vdots$	...	...	...	...	$\vdots$
O_n	c_{n1}	c_{n2}	...	c_{nm}	b_n
demandas	d_1	d_2	...	d_m	

Tabela 1: Tabela de Transporte

Transporte aplicado em RDC pode ser dividido em três passos:

Passo 1: Na célula do canto superior esquerdo da tabela (i, r) e atribuir o valor máximo possível a variável $x^r_{ij(r)}(x^r_{ij(r)} = min(b_i, d_r))$, se o servidor O_i possui o conteúdo da requisição r.

Passo 2: Seja b'_i, o valor da banda disponível no servidor i e d'_r, o valor da banda do conteúdo da requisição r ainda não alocada. Inicialmente $b'_i = b_i$ e $d'_r = d_r$. Se

- $d'_r > b'_i$, a largura de banda do servidor O_i está esgotada, mas a requisição do cliente D_r não é totalmente atendida, $d'_r = d'_r - b_i$ e $b'_i = 0$. Mover, se possível, para a segunda linha $(i+1, r)$ e atribuir a $x^r_{i+1j(r)}$ o $min(b'_{i+1}, d'_r)$, se o servidor O_{i+1} possui o conteúdo da requisição r, excluindo da tabela o servidor O_i.

- $b'_i > d'_r$, a requisição do cliente D_r é totalmente satisfeita, mas a largura de banda do servidor O_i não é totalmente utilizada, $b'_r = b'_r - d'_r$. e $d'_r = 0$. Mover, se possível, para a próxima coluna $(i, r+1)$ e atribuir a $x^{r+1}_{ij(r+1)}$ o $min(b'_i, d'_{r+1})$, excluindo da tabela a requisição D_r.

- $d'_r = b'_i$, a largura de banda do servidor O_i está esgotada e a requisição do cliente D_r é totalmente atendida, $d'_r = b'_i = 0$. Exclui-se o servidor O_i e a requisição D_r. Caso ainda haja requisições e servidores na tabela, retorne ao passo 1. Caso contrário, terminar o algoritmo.

Passo 3: Caso ainda haja requisições e servidores para compor a tabela, retorne ao passo 2. Caso contrário, terminar o algoritmo.

O método do Custo Mínimo é similar ao do Canto Noroeste. Eles diferem na escolha das células. O Canto Noroeste escolhe sempre a célula mais esquerda superior e o Custo Mínimo, a célula que possui o menor custo de comunicação.

5 Algoritmo Distribuído

Este trabalho propõe uma heurística distribuída baseada nos métodos do Canto Noroeste e Custo Mínimo para resolver o PAC. O algoritmo resolve a versão *off-line* do problema, ou seja, as requisições por conteúdos são informações de um dado instante de tempo. O problema também pode ser utilizado para a avaliação da rede. As requisições são extraídas durante um intervalo de tempo, permitindo obter a estatística do número de requisições por um determinado conteúdo. A estatística pode, então, compor os dados de entrada do problema. Assim, é possível obter informações importantes da rede, como a carga nos servidores e o custo total de transporte dos conteúdos, possibilitando melhorias na QoS.

5.1 Pseudocódigo

Nesta seção é feita a descrição do algoritmo distribuído baseado no método do Custo Mínimo. Para isto, algumas considerações são feitas e algumas notações são usadas. Como é preferível que um cliente acesse a cópia de um conteúdo através do servidor mais próximo, as requisições dos clientes foram, então, colocadas nestes servidores. Com isso, a figura do cliente foi retirada e toda a responsabilidade da distribuição de requisições foi dada aos servidores, o que de fato acontece em uma RDC. Assim, cada servidor possui uma lista de requisições e as informações das posições das réplicas dos conteúdos em todos os servidores. Os servidores também conhecem os custos de comunicação com todos os outros servidores. Cada servidor possui as seguintes variáveis:

$meuId$	id associado ao servidor
$idMenor$	servidor mais próximo
S	Conjunto de Servidores
$lCont$	Lista de Conteúdos locais
$nReq$	Número de Requisições
$lReq$	Lista de Requisições
req_r	requisição r
$cont_r$	id. do conteúdo da requisição r
$bandCont_r$	banda exigida pela requisição r
$quantAlocada_r$	banda alocada para requisição r
$lReqNalocadas$	lista de requisições não alocadas
$banda$	largura de banda
$atendeu_{ic}$	banda alocada pelo servidor devido pedido do servidor i pelo conteúdo c
$atendimento_{ri}$	banda para requisição r atendida pelo servidor i

Cada servidor, inicialmente, tenta alocar cada uma das suas requisições locais. Uma mensagem $ALOCA$ é enviada ao servidor de menor custo que possui o conteúdo associado a requisição (Algoritmos 1 e 2). Um servidor ao receber a mensagem $ALOCA$ (Algoritmo 3), verifica a disponibilidade de banda. Se for possível atender a requisição, uma mensagem ACK é enviada ao servidor que requisitou o conteúdo, caso contrário, uma mensagem $NACK$ é enviada.

Algoritmo 1: Inicialização

para r ***de*** 1 ***até*** $nReq$ **faça**
  se $cont_r \in lCont$ **então**
    $tentaAlocar(r, bandCont_r, cont_r)$
  senão
    $idMenor-$ id do servidor mais próximo ainda não verificado que possui $cont_r$;
    envia $ALOCA(r, bandCont_r, cont_r)$ para $idMenor$;

Algoritmo 2: tentaAlocar (r, tam, c)

se $banda > 0$ **então**
  se $tam <= banda$ **então**
    $banda := banda - tam$; **e** $atendeu_{meuIdc} := tam$;
    $atendimento_{rmeuId} := tam$; **e** $quantAlocada_r := quantAlocada_r + tam$;
  senão
    $atendeu_{meuIdc} := banda$; **e** $atendimento_{rmeuId} := banda$;
    $quantAlocada_r := quantAlocada_r + banda$; **e** $banda := 0$;
se *($quantAlocada_r < bandCont_r$) e ($\exists$ servidor não verificado que possui c)* **então**
  $idMenor-$ id do servidor mais próximo ainda não verificado que possui c;
  envia $ALOCA(r, bandCont_r - quantAlocada_r, c)$ para $idMenor$;

Um servidor ao receber um ACK (Algoritmos 4 e 6), desconta a quantidade que já foi atendida. Se esta quantidade não representar a banda total do conteúdo, uma mensagem $ALOCA$ é enviada a outro servidor que ainda não foi verificado (i.e. o próximo servidor com menor custo). No entanto, um servidor ao receber um $NACK$ (Algoritmo 5), apenas envia $ALOCA$ a outro servidor não verificado. O algoritmo termina quando todos os servidores conseguirem atribuir todas as suas requisições a um servidor. Entretanto, uma requisição pode não ser atribuída a nenhum servidor local. Isto acontece porque esses servidores já usaram a sua banda para atender requisições dos outros conteúdos. Quando esta situação acontece um mecanismo de troca é iniciado. O

Algoritmo 3: Ao receber uma mensagem $ALOCA(r, tam, c)$ do servidor i

se $banda > 0$ **então**
  se $tam <= banda$ **então**
    $banda := banda - tam$; **e** $atendeu_{ic} := tam$;
    envia $ACK(r, tam, c)$ para servidor i;
  senão
    $atendeu_{ic} := banda$;
    envia $ACK(r, banda, c)$ para servidor i;
    $banda := 0$;
senão
  envia $NACK(r, tam, c)$ para servidor i;

mecanismo de troca é realizado quando mensagens de alocação de uma requisição foram enviadas a todos os servidores e não foi possível atendê-la totalmente. Este mecanismo consiste em enviar uma mensagem $TROCA$ ao servidor de menor custo que possui o conteúdo da requisição. Este servidor, por sua vez, ao receber a mensagem deve alocar o restante da banda do conteúdo ainda não atendido. Para isto, ele deve retirar requisições para as quais a quantidade de banda alocada a elas permite o novo atendimento e que não tenham sido atendidas pelo mecanismo de troca (não foram trocadas) (Algoritmo 11). Uma vez que uma requisição foi trocada de servidor, esta não poderá ser trocada novamente. Isto é feito para evitar que o algoritmo entre em *loop*. Caso não seja possível retirar nenhuma requisição ou as requisições retiradas não sejam suficientes para alocar o novo atendimento, uma mensagem $NACKTROCA$ (Algoritmo 9) é enviada para o servidor que enviou a mensagem $TROCA$ com a quantidade de banda ainda não alocada para que um novo pedido seja enviado a outro servidor de menor custo depois do que já foi escolhido. No entanto, se não for possível enviar $TROCA$ a outro servidor, ou seja, a mensagem $TROCA$ já foi enviada para todos os servidores, a requisição é classificada como "atendida pelo servidor central" e um custo alto é adicionado para representar que a requisição é atendida pelo servidor central. O servidor central consiste do servidor origem do conteúdo requisitado. Caso consiga retirar requisições, mensagens $TSERVIDOR$ são enviadas aos servidores responsáveis pelas requisições retiradas, para que estes possam encontrar outros servidores que atendam estas requisições (Algoritmo 10).

As linhas 1 a 19 do Algoritmo 10 referem-se à busca por requisições referentes ao conteúdo

Algoritmo 4: Ao receber uma mensagem $ACK(r, tam, c)$ do servidor i

$procedimentoAck(r, tam, c, i)$;

que deixou de ser atendido e que não foram atendidas pelo mecanismo de troca. Ao encontrar a requisição, esta é deixada de ser atendida e uma mensagem $ALOCA$ é enviada ao servidor mais próximo ainda não verificado diferente do servidor que enviou a mensagem $TSERVIDOR$ (linha 14). Caso não exista nenhum servidor não verificado, uma mensagem $TROCA$ é enviada ao servidor mais próximo diferente do servidor que enviou a mensagem $TSERVIDOR$ (linha 17). As linhas 20 a 24 do Algoritmo 10 referem-se ao envio de um ACK para a confirmação de alocação para o servidor que originou o mecanismo de troca. A linha 26 refere-se ao envio da mensagem

$NACKTSERV$ para que encontre outro servidor que não tenha atendido alguma requisição por meio do mecanismo de troca.

Algoritmo 5: Ao receber uma mensagem $NACK(r, tam, c)$ do servidor i

se $\exists$ *servidor não verificado que possui* c **então**
  $idMenor-$ id do servidor mais próximo ainda não verificado que possui c;
  envia $ALOCA(r, bandCont_r - quantAlocada_r, c)$ para $idMenor$;
senão
  $idMenor-$ id do servidor mais próximo que possui c;
  envia $TROCA(r, c, tam)$ para $idMenor$;

Algoritmo 6: procedimentoAck(r, tam, c, i)

$atendimento_{ri} := tam$; **e** $quantAlocada_r := quantAlocada_r + tam$;
se $quantAlocada_r < bandCont_r$ **então**
  se $\exists$ *servidor não verificado que possui* c **então**
    $idMenor-$ id do servidor mais próximo ainda não verificado $\neq i$ que possui c;
    envia $ALOCA(r, bandCont_r - quantAlocada_r, c)$ para $idMenor$;
  senão
    $idMenor-$ id do servidor mais próximo que possui c;
    envia $TROCA(r, c, tam)$ para $idMenor$;

Algoritmo 7: Ao receber uma mensagem $TROCA(r, c, tam)$ do servidor i

$procedimentoTroca(r, c, tam, i)$;

5.2 Complexidades de Mensagens e Tempo

Quando um servidor tenta alocar uma requisição, no pior caso, ele envia uma mensagem de tentativa de alocação a todos os outros servidores que contenha o conteúdo requisitado. Para este caso, n mensagens são enviadas, onde n é o número de servidores. Logo, para o atendimento de todas as requisições, o número total de mensagens enviadas é da ordem de $n.m$, onde m é o número de requisições. No entanto, caso não seja possível alocar uma requisição a nenhum servidor, uma operação de troca é iniciada. Uma operação de troca conforme descrita na Seção 5.1 pode ser muito custosa, pois pode desencadear outras operações de troca. Isso pode resultar na reatribuição de todas as requisições entre os servidores, o que resulta no envio de mais $n.m$ mensagens. Com isso, a complexidade de mensagens é $O(n.m)$. A complexidade de tempo global em modelos distribuídos assíncronos assume que as computações locais não gastam tempo, e que o tempo para enviar uma mensagem para um nó vizinho é $O(1)$. A complexidade de tempo global é então definida como o número de mensagens na maior cadeia causal do tipo "recebeu uma mensagem e enviou uma mensagem como consequência" ocorrida durante a execução do algoritmo. Ao tentar alocar uma requisição, suponha que seja necessário enviar mensagens de tentativa de alocação a todos os servidores que contenham o conteúdo, resultando no envio de n mensagens. Entretanto, quando não for possível atender a essa requisição, uma operação de troca é iniciada, desencadeando a reatribuição de todas as outras requisições. Assim, além das n mensagens enviadas, outras n mensagens serão usadas para tentar realocar cada requisição, no total de m requisições. Portanto, a complexidade de tempo global é $O(n.m)$. A complexidade de tempo local refere-se ao tempo de realizar computações locais em um nó entre o recebimento de duas mensagens consecutivas. Como pode ser verificado nos algoritmos da Seção 5.1, o tempo local gasto por cada nó i (servidor i) é $O(r_i)$, onde r_i é número de requisições do servidor i.

Algoritmo 8: $procedimentoTroca(r, c, tam, s)$

$tamAux := tam$;
enquanto $tamAux > 0$ **faça**
  seleciona um servidor $s' \in S$ não selecionado em que $atendeu_{s'c'} > 0$, para $c'! = c$;
  se *não selecionou nenhum servidor* **então**
    envia $NACKTROCA(r, c, tamAux)$ para o servidor s;
    exit;
  se $atendeu_{s'c'} >= tamAux$ **então**
    $atendeu_{sc} := atendeu_{sc} + tamAux$;
    $atendeu_{s'c'} := atendeu_{s'c'} - tamAux$; **e** $tamAux := 0$;
  senão
    $tamAux := tamAux - atendeu_{s'c'}$; **e** $tam := atendeu_{s'c'}$;
    $atendeu_{sc} := atendeu_{sc} + atendeu_{s'c'}$; **e** $atendeu_{s'c'} := 0$;
  envia $TSERVIDOR(r, c', tam, s)$ para o servidor s';

Algoritmo 9: Ao receber uma mensagem $NACKTROCA(r, c, tam)$ do servidor i

$idMenor-$ id do servidor mais próximo depois de i que possui c;
se $idMenor \neq vazio$ **então**
  envia $TROCA(r, c, tam)$ para $idMenor$;
senão
  $r :=$ "atendida pelo servidor central";

Algoritmo 10: Ao receber uma mensagem $TSERVIDOR(r, c, tam, s)$ do servidor i

$tamAux := tam$;
para r' ***de*** 1 ***até*** $nReq$ **faça**
  se *(*$cont_{r'} = c$*) e (*$atendimento_{r'i} > 0$*) e (*r' *não foi trocada)* **então**
    se $atendimento_{r'i} >= tamAux$ **então**
      $quantAlocada_{r'} := quantAlocada'_r - tamAux$;
      $atendimento_{r'i} := atendimento_{r'i} - tamAux$;
      $q = tamAux$; **e** $tamAux = 0$;
    senão
      $quantAlocada_{r'} := quantAlocada'_r - atendimento_{r'i}$;
      $tamAux = tamAux - atendimento_{r'i}$;
      $atendimento_{r'i} := 0$; **e** $q = atendimento_{r'i}$;
    se $\exists$ *servidor não verificado que possui* c **então**
      $idMenor-$ id do servidor mais próximo ainda não verificado $\neq i$ que possui c;
      envia $ALOCA(r, q, c)$ para $idMenor$;
    senão
      $idMenor-$ id do servidor mais próximo $\neq i$ que possui c;
      envia $TROCA(r, c, q)$ para $idMenor$;
    se $tamAux = 0$ **então**
      exit;
se $tamAux = 0$ **então**
  se *não sou* s **então**
    envia $ACK(r, tam, c)$ para servidor s;
  senão
    $procedimentoAck(r, tam, c, i)$;
senão
  envia $NACKTSERV(r, c, tamAux, s)$ para servidor i;

Algoritmo 11: Ao receber uma mensagem $NACKTSERV(r, c, tam, s)$ do servidor i

$atendeu_{sc} := atendeu_{sc} - tam$; **e** $atendeu_{ic'} := atendeu_{ic'} + tam$;
$procedimentoTroca(r, c, tam, s)$;

O algoritmo distribuído proposto tem, no pior caso, complexidade de $O(n.m)$ mensagens, $O(n.m)$ tempo global e $O(r_i)$ tempo local. Uma alternativa para resolver o PAC seria a eleição de um nó (servidor) líder para resolver localmente o problema e depois propagar a informação. Mas como as informações na RDC estão distribuídas e nenhum servidor na RDC conhece todos os pedidos dos clientes de todos os outros servidores locais, há um custo para centralizar a informação. A propagação de informação neste caso possui complexidade de mensagens de $O(n^2)$ (com tamanho correspondente as listas de requisições) e $O(n)$ de tempo global. Além disso, a complexidade de tempo do algoritmo sequencial do Canto Noroeste e do Custo Mínimo aplicado ao PAC é $O(n.m)$, pois no método sequencial a Tabela de Transporte é percorrida, esgotando a banda dos servidores e satisfazendo as requisições por completo. Note que quando um servidor (s) esgota sua banda satisfazendo por completo algumas requisições ($O(m)$) e quando existe um servidor com banda disponível e requisições não satisfeitas, deve-se continuar o procedimento e escolher um próximo servidor para continuar a percorrer a tabela. Assim, para cada requisição satisfeita por s, percorre-se os servidores que possuem o conteúdo requisitado até encontrar um servidor com banda disponível ($O(max_r.n)$ onde max_r é número máximo de requisições atendidas por um servidor). Logo, para um servidor percorrido escolher outro para continuar, no pior caso, m requisições são percorridas e para cada uma destas satisfeitas por ele, n servidores são percorridos ($O(m + max_r.n)$, onde $max_r << m$). Como se deve esgotar a banda de todos os servidores, os n servidores devem ser percorridos, resultando em complexidade de tempo $O(n.m)$. No entanto, se uma requisição não foi atendida, todos os servidores são percorridos para encontrar um que seja capaz de realizar a troca de atendimento de uma requisição por essa. Com isso, a complexidade de tempo do algoritmo sequencial aumentaria para $O(n^2.m)$. Considerando essas complexidades e a demanda de memória do nó líder, a abordagem distribuída proposta é uma alternativa atraente para o problema.

6 Resultados Experimentais

Para validar o algoritmo proposto, os testes foram realizados em um computador AMD Phenom x4 2.3GHz com 4GBytes de memória e sistema operacional Linux. O algoritmo foi implementado em ANSI C e MPICH2 para simulação distribuída com troca de mensagens. Devido ao método do Custo Mínimo apresentar melhores soluções, em média, do que o método do Canto Noroeste, somente os resultados do algoritmo DistPAC baseado no método do Custo Mínimo são apresentados. Os valores para o método do Custo Mínimo sequencial também foram obtidos, mas como foram encontrados resultados similares ao distribuído, estes valores não foram apresentados. As instâncias utilizadas nos testes foram obtidas de Neves *et al*. (2010). As instâncias são divididas em três classes: fáceis de resolver, intermediárias e difíceis. Para cada classe, há 20 instâncias, 5 para cada número de servidores, 10, 20, 30 e 50. Os números de requisições totais das instâncias com 10, 20, 30 e 50 servidores são, em média, 550, 1200, 2050 e 3300, respectivamente. Além disso, a largura de banda exigida pelas requisições não está disponível nessas instâncias, assim, a largura de banda considerada foi 15% do tamanho do conteúdo associado a requisição. As instâncias consideram que as requisições dos clientes estão nos servidores mais próximos a ele. Assim, cada servidor possui uma lista de requisições.

Para comparar os resultados obtidos pelo algoritmo, a formulação matemática (da Seção 3) foi resolvida para encontrar a solução ótima. Para isto, o resolvedor $CPLEX11.2$ foi utilizado. Como descrito na Seção 5.1, as posições das réplicas são dados de entrada para o algoritmo. Estes dados também não estão disponíveis nessas instâncias. Assim, esses dados foram gerados através do posicionamento ótimo obtido por formulação matemática com o uso do CPLEX. Cada instância foi executada 10 vezes e calculada a média das execuções para as soluções encontradas, o número de mensagens trocadas e o tempo global. Estes dados estão na Tabela 2, juntamente com o valor

da solução ótima, o percentual da distância do ótimo e do desvio padrão. O algoritmo obteve bons resultados, ficando, em média, 1.6% do ótimo e apresentando desvio padrão menor que 2.4%. A instância que apresentou pior solução fica apenas 5% distante do ótimo. E apesar da complexidade do número de mensagens ser alta devido ao mecanismo de troca, em média, ocorreram poucas trocas, o que diminui esse número de mensagens na prática. Isto também justifica o tempo global relativamente baixo em relação a sua complexidade. Nas instâncias consideradas fáceis, observa-se o valor de solução igual a zero. Isto ocorre devido à existência de réplicas de todos os conteúdos em todos os servidores, o que permite que para estas instâncias, independendo do número de servidores, o algoritmo encontre sempre o valor ótimo.

7 Conclusão

Este trabalho descreve o Problema de Atribuir Clientes a servidores em uma Rede de Distribuição de Conteúdos, apresenta uma formulação matemática para versão *off-line* do problema e propõe uma abordagem distribuída para o problema baseada nos métodos do Canto Noroeste e do Custo Mínimo para obtenção de uma solução inicial viável para o Simplex de Transporte. O algoritmo proposto apresentou resultados próximos aos ótimos, obtendo, em média, soluções distantes 1.5% do ótimo. Como trabalho futuro, a versão *on-line* do problema, representando melhor a realidade de uma rede, será apresentada e implementada. Além disso, um estudo está sendo feito sobre o Simplex de Transporte, para que seja possível sua implementação distribuída.

Referências

Ahuja, R. K.; Magnanti, T. L.; Orlin, J. B. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall. ISBN 013617549X, 1993.

Bektas, T.; Cordeau, J.-F.; Erkut, E.; Laporte, G. (2008). Exact algorithms for the joint object placement and request routing problem in content distribution networks. *Comput. Oper. Res.*, v. 35, n. 12, p. 3860–3884.

Bektas, T.; Oguz, O.; Ouveysi, I. (2007). Designing cost-effective content distribution networks. *Comput. Oper. Res.*, v. 34, n. 8, p. 2436–2449.

Dantzig, G. Application of the simplex method to a transportation problem. Koopmans, T. (Ed.), *Activity analysis of production and allocation*, p. 359–373. J. Wiley, New York, 1951.

Hall, J. (2010). Towards a practical parallelisation of the simplex method. *Computational Management Science*, v. 7, p. 139–170.

Mundur, P.; Arankalle, P. (2006). Optimal server allocations for streaming multimedia applications on the internet. *Computer Networks*, v. 50, n. 18, p. 3608 – 3621.

Neves, T. A.; de A. Drummond, L. M.; Ochi, L. S.; Albuquerque, C.; Uchoa, E. (2010). Solving replica placement and request distribution in content distribution networks. *Electronic Notes in Discrete Mathematics*, v. 36, p. 89–96.

Shah, S.; Ramamritham, K.; Ravishankar, C. Client assignment in content dissemination networks for dynamic data. *Proceedings of the 31st international conference on Very large data bases*, p. 673–684. ISBN 1-59593-154-6, 2005.

Zhou, X.; Xu, C.-Z. (2007). Efficient algorithms of video replication and placement on a cluster of streaming servers. *Journal of Network and Computer Applications*, v. 30, n. 2, p. 515 – 540.

Tabela 2: Resultados

Instâncias	Ótimo	DistPAC	Mensagens	Tempo Global	Distância do Ótimo(%)	Desvio Padrão(%)
inst_10_1	0.0	0.0	108.0	3.0	0.00	0.00
inst_10_2	0.0	0.0	108.0	3.0	0.00	0.00
inst_10_3	0.0	0.0	108.0	3.0	0.00	0.00
inst_10_4	0.0	0.0	108.0	3.0	0.00	0.00
inst_10_5	0.0	0.0	108.0	3.0	0.00	0.00
inst_10_6	471763.2	471763.2	446.0	34.1	0.00	0.00
inst_10_7	306226.7	306249.5	347.2	28.3	0.01	0.01
inst_10_8	378230.8	379434.9	382.8	29.4	0.32	0.01
inst_10_9	461161.2	471943.3	534.2	31.0	2.34	1.38
inst_10_10	501081.9	513089.5	476.8	27.3	2.40	2.00
inst_10_11	471763.2	471763.2	446.0	34.3	0.00	0.00
inst_10_12	316065.0	317246.9	363.8	24.7	0.37	0.59
inst_10_13	378230.8	379419.2	383.6	29.8	0.31	0.01
inst_10_14	461161.2	477054.6	490.3	26.0	3.45	2.34
inst_10_15	501081.9	517749.4	476.6	27.6	3.33	2.37
inst_20_1	0.0	0.0	418.0	3.0	0.00	0.00
inst_20_2	0.0	0.0	418.0	3.0	0.00	0.00
inst_20_3	0.0	0.0	418.0	3.0	0.00	0.00
inst_20_4	0.0	0.0	418.0	3.0	0.00	0.00
inst_20_5	0.0	0.0	418.0	3.0	0.00	0.00
inst_20_6	1434569.8	1484250.0	1551.2	41.8	3.46	0.78
inst_20_7	916305.8	954801.2	1293.2	38.2	4.20	1.21
inst_20_8	1158373.3	1167266.2	1441.0	34.0	0.77	0.32
inst_20_9	1160990.9	1183816.9	1499.6	42.1	1.97	0.66
inst_20_10	853256.2	864462.5	1123.4	33.0	1.31	0.51
inst_20_11	1434569.8	1483717.4	1562.0	42.7	3.43	1.06
inst_20_12	916305.8	946151.4	1284.4	40.9	3.26	1.08
inst_20_13	1158373.3	1166845.0	1454.5	35.1	0.73	0.28
inst_20_14	1160990.9	1184041.8	1884.5	70.4	1.99	0.64
inst_20_15	839075.6	848217.3	1092.8	31.2	1.09	0.23
inst_30_1	0.0	0.0	928.0	3.0	0.00	0.00
inst_30_2	0.0	0.0	928.0	3.0	0.00	0.00
inst_30_3	3192.4	3192.4	980.0	32.8	0.00	0.00
inst_30_4	0.0	0.0	928.0	3.0	0.00	0.00
inst_30_5	0.0	0.0	928.0	3.0	0.00	0.00
inst_30_6	1443886.8	1467822.1	2149.6	35.3	1.66	0.39
inst_30_7	1280967.4	1305033.3	199.4	29.9	1.88	0.19
inst_30_8	1132308.9	1184228.7	1908.4	34.0	4.59	1.55
inst_30_9	1169034.7	1224238.7	2013.8	37.7	4.72	1.63
inst_30_10	1150970.7	1181051.3	2015.9	37.9	2.61	0.56
inst_30_11	1458770.6	1485796.8	2165.4	36.2	1.85	0.38
inst_30_12	1280967.4	1304256.7	1999.6	30.5	1.82	0.62
inst_30_13	1132308.9	1172052.1	1894.0	32.6	3.51	1.03
inst_30_14	1182624.5	1236735.0	2024.4	36.4	4.58	1.20
inst_30_15	1150970.7	1179446.5	2048.5	40.1	2.47	0.62
inst_50_1	0.0	0.0	2548.0	3.0	0.00	0.00
inst_50_2	23162.4	23162.4	2892.0	302.0	0.00	0.00
inst_50_3	34877.9	34877.9	2938.0	339.0	0.00	0.00
inst_50_4	0.0	0.0	2548.0	3.0	0.00	0.00
inst_50_5	49511.3	49511.3	3085.0	487.0	0.00	0.00
inst_50_6	2223511.6	2335412.3	4536.6	39.1	5.03	0.98
inst_50_7	1655211.8	1670543.2	4046.6	32.0	0.93	0.26
inst_50_8	3065126.1	3131303.9	4931.9	43.1	2.16	0.72
inst_50_9	3029900.9	3124683.8	5022.8	42.8	3.13	0.62
inst_50_10	2196471.3	2231851.8	4316.0	39.7	1.61	0.45
inst_50_11	2223511.6	2331908.5	4529.2	39.5	4.88	0.87
inst_50_12	1655211.8	1666422.2	4040.6	31.3	0.68	0.41
inst_50_13	3065126.1	3138199.1	5102.4	44.1	2.38	0.77
inst_50_14	3011459.5	3091734.2	5034.4	43.1	2.67	0.39
inst_50_15	2196403.0	2218612.3	4298.4	36.4	1.01	0.26