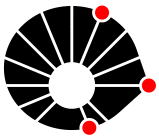

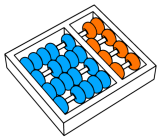

Análise quantitativa de caso para Servidores em Fog x Cloud: Procurando situações vantajosas para um Servidor mais próximo do Usuário

B. Itiroko *G. Pereira* *T. Jun* *L. Pagnez*
L. F. Bittencourt

Relatório Técnico - IC-PFG-19-43
Projeto Final de Graduação
2019 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Análise de Servidores em Fog

Bianca Itiroko* Giovani Nascimento† Thomas Jun‡ Leonardo Pagnez§

Resumo

Este trabalho tem por objetivo realizar estudos na área de computação *Fog*, que é caracterizada pela distribuição de processamentos em bordas da rede. Isso foi feito de forma prática, isto é, desenvolveu-se uma aplicação que envolve processamento de vídeo e de arquivos de áudio e distribuiu-se em uma máquina próxima (localizada na UNICAMP) e em outra situada no estado da Carolina do Sul nos Estados Unidos.

Dessa forma, foi possível observar como o Hardware das máquinas, que define sua capacidade de processamento, e o delay de comunicação entre as máquinas envolvidas afetam o tempo total para que o processamento desejado seja realizado.

Com isso foram definidos diferentes cenários de estudo variando máquinas alocadas em diferentes serviços de Cloud, utilizando diferentes arquivos de vídeo como objeto de processamento variando em tamanho e duração dos clipes, e variando os serviços envolvidos utilizando sistemas externos para armazenamento dos conteúdos de vídeo, como o YouTube.

Ao final das análises, foi possível notar as vantagens e desvantagens de cada cenário estabelecido, bem como compreender melhor qual alocação de máquinas e serviços pode ser mais vantajosa dadas diferentes situações de uso.

1 Introdução

1.1 Motivação

À medida que a tecnologia evolui e se torna disponível para cada vez mais pessoas, a necessidade de evoluir serviços em termos de escala, disponibilidade e velocidade se torna cada vez maior. Isso implica em uma necessidade imprescindível em melhorar e otimizar continuamente a forma como pessoas utilizam a internet e como se conectam com o mundo.

Além disso, concentrar processamentos em contextos de escopo local acaba por tomar um caminho contrário à evolução de escala e velocidade.

Portanto, existem algumas estratégias para resolver esses problemas, sendo que uma das mais utilizadas é a computação em nuvem, em que são delegadas atividades para uma máquina remota, tornando viável a computação de larga escala em um data center.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
‡Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
§Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

Ter diferentes dispositivos conectados a uma mesma fonte de verdade possibilita uma grande dinamicidade em relação aos dados que devem alimentar os serviços locais, isto é, a comunicação torna-se mais confiável e rápida.

Um dos fatores que aumenta significativamente o tráfego de dados é a internet das coisas, visto que são equipamentos que geram e necessitam de dados constantemente. Estima-se que até 2020, existirão mais de 50 bilhões de dispositivos conectados, como cita Chase et al. [1].

Entretanto, à medida que ocorre o aumento no volume de requisições a serviços remotos em nuvem, o tempo de latência - no caso, envolvendo tanto o fator distância quanto poder de processamento - torna-se um problema.

Surge então um contexto específico com necessidades de conexão específicas: para os dispositivos relacionados a internet das coisas, os dados são produzidos e consumidos próximos a borda - isto é, próximo ao destino final. Assim, a descentralização da fonte de dados se torna a solução otimizada e que auxilia em concentrar os processamentos num ponto específico (possivelmente mais distante) apenas nos casos necessários.

Por isso, a possibilidade de ter esses processamentos em regiões de borda da rede tem sido amplamente estudadas e recebe o nome de computação em névoa (fog computing).

Uma das áreas que se beneficia do poder de processamento remoto é a de mídia. Presente na área de segurança, entretenimento, educacional, entre muitas outras, a análise de mídias, como vídeos e áudios, é algo muito custoso e que exige poder de processamento [2].

Com o desenvolvimento cada vez maior do conceito de Smart Cities, torna-se necessário extrair e processar eficientemente dados para que possam ser extraídos novos padrões de comportamento e possíveis soluções para problemas do cotidiano. Nesse caso é possível ver uma clara razão para utilizar o fog computing: trata-se de uma situação muito dirigida pelo local em que ocorre e o tráfego de dados não precisa se disseminar muito além dali, podendo então mandar para a cloud apenas o que for essencial.

Realizar esse processamento em uma região de borda da rede otimizaria tanto o tempo de latência quanto a seleção de dados.

1.2 Objetivos

Tendo em vista as motivações citadas anteriormente, o intuito deste projeto é estudar o processamento de conteúdos de mídia em serviços remotos, como em infraestruturas de nuvem e névoa, no caso o objeto de estudo escolhido foram de arquivos de vídeo.

Com isso, é possível definir diferentes ambientes que irão executar a tarefa de processamento do vídeo, e analisar esses processos a fim de compreender melhor quais são os parâmetros que fazem com que o processamento remoto do conteúdo seja mais vantajoso.

2 Conceitos Básicos

2.1 Fog Computing

O conceito de **Fog Computing** consiste em realizar processamentos requeridos por um cliente não necessariamente todo na cloud, mas tornar possível isso ser realizado em servidores

localizados nas bordas da rede. A principal proposta consiste na utilização de Cloudlets: dispositivos que se localizam fisicamente próximos e realizam parte do processamento da Cloud, sendo que estão mais perto de quem requisitou a informação, de forma a otimizar os tempos de comunicação [3].

2.2 HTTP

HTTP (*Hypertext Transfer Protocol*) é um protocolo de comunicação de rede e é a base das comunicações utilizado na internet atualmente. Ele funciona dentro de um modelo computacional de *cliente-servidor* onde uma das partes faz uma requisição HTTP (cliente) e a outra parte recebe essa requisição e envia uma resposta (servidor). Um exemplo comum é um site web, onde o servidor fornece recursos como conteúdos de mídia ou arquivos HTML conforme o cliente os requisita para serem apresentados [9].

O protocolo HTTP foi o protocolo utilizado neste projeto para transferir arquivos entre máquinas distintas através da internet. Na maior parte dos casos, o protocolo foi abstraído por uma biblioteca, como o Wget.

2.3 Cloud Compting

Cloud Computing, ou computação em nuvem, é a entrega de diversos serviços e recursos computacionais, tais como armazenamento de dados e poder de processamento, através da Internet.

Existem três tipos de serviços de Cloud Computing, sendo eles IaaS, PaaS, SaaS.

SaaS são os serviços de aplicações em nuvem, que usam a web para acessar aplicativos gerenciados por terceiros com uma interface executada no lado do cliente. Já o PaaS, ou *Platform as a Service*, envolve um ambiente virtual para criação, hospedagem e controle de softwares e bancos de dados. As empresas pagam pelo serviço e é transferido ao fornecedor a responsabilidade de manutenção do mesmo.

O IaaS, ou *Infrastructure as a Service*, é o provisionamento de hardwares ou máquinas virtuais. Nesse caso, empresas são contratadas para disponibilizar uma capacidade de hardware de acordo com as necessidades de quem a contrata (memória, armazenamento, processamento, entre outros).

Sua principal vantagem é a escalabilidade, como trata-se de um serviço prestado, as configurações das máquinas contratadas são totalmente mutáveis e muito mais flexível do que quando fala-se de questões de hardware.

Para o desenvolvimento do projeto, foram utilizados dois exemplos de IaaS: o Google Cloud e o IC Cloud. Tinha-se como necessidade mover processamentos do ambiente local para o remoto, além de explorar outras localizações geográficas para os servidores.

3 Ferramentas e Serviços Utilizados

3.1 Wget

Wget é um pacote, de código aberto, para realizar a transferência de arquivos através de comunicações com os protocolos de rede HTTP, HTTPS, FTP e FTPS, que são os protocolos

de comunicação de rede mais utilizados. O Wget pode ser utilizado como uma ferramenta de linha de comando, não interativa, o que permite que seja utilizada facilmente em scripts para realizar transferência de arquivos [4].

Esse pacote foi utilizado como o método de transferência dos arquivos de vídeo entre máquinas distintas, o que permitiu que os arquivos de vídeo fossem enviados a servidores remotos para que pudessem ser processados.

3.2 MoviePy

MoviePy é uma biblioteca open source, desenvolvida em Python, para o processamento de arquivos de vídeo. Ela contém operações básicas para tratar clipes de vídeo, como cortes, concatenações, inserções de imagens ou textos nos clipes. A biblioteca também é capaz de abrir os formatos mais comuns de arquivos de vídeo [5].

A parte utilizada da biblioteca, além de abrir os arquivos de vídeo para edição, foi a função de extração da faixa de áudio dos clipes de vídeo.

3.3 Pytube

Pytube é um framework escrito em Python, que pode ser utilizado tanto como uma biblioteca em scripts Python como comando diretamente num terminal, para download de clipes de áudio do serviço do YouTube.

A biblioteca permite o download de clipes de vídeo do YouTube em diversos formatos e resoluções, basicamente com todos os formatos nos quais o vídeo em questão está disponível no YouTube [6].

Esta biblioteca foi utilizada no projeto para permitir o download de vídeos do YouTube, que foi um dos casos de teste criados para análise.

3.4 Google Cloud

O Google Cloud Platform é o nome dado a um conjunto de serviços oferecidos pela Google (companhia de serviços e produtos web e de tecnologia) que utilizam a mesma infraestrutura que a própria empresa utiliza para seus produtos disponibilizados para os usuários (como o seu serviço de buscador Google ou o YouTube).

Os serviços disponibilizados pelo Google Cloud Platform são diversos como: armazenamento de dados, computação, treinamento de redes neurais, bancos de dados, aprendizados de máquina, entre outros [7].

Foi utilizado o serviço Compute Engine do Google Cloud, que oferece máquinas virtuais com inicialização rápida e com desempenho razoável fazendo uso de SSD local com capacidade de persistência [8].

O serviço disponibiliza uma máquina gratuitamente por até 1 ano que pode ser utilizada com finalidades acadêmicas. Para a realização das análises, foi utilizado uma instância do tipo n1-standard-1 (1 vCPU, 3,75 GB de memória).

3.5 Cloud do IC

Utiizado como o servidor em Fog para usuários da Unicamp e para este estudo de caso, Cloud IC é uma solução em Cloud para o vasto espaço digital do Instituo de Computação da Unicamp. Tal solução foi baseado na tecnologia do OpenStack disponibilizada na Internet para uso público.

Dotada de diversas configurações padrões, membros da comunidade que se interessarem em disponibilizar suas Aplicações no Cloud podem utilizar das diversas configurações padrões do Ambiente, chamadas de Flavors, e disponibilizar seu IP-fixo para acessos remotos via SSH e HTTP, levantando assim um Servidor Web em rápidos passos.

As Instâncias levantadas no Cloud do IC pelos membros deste Projeto tinham os seguintes limites: Até 16 CPU virtuais, 32 gb de RAM e até 100 GB de armazenamento.

3.6 AWS

O AWS (Amazon Web Services) é uma plataforma de serviços da Amazon (empresa norte-americana de comércio eletrônico) em nuvem que fornece diversos serviços, dentre eles bancos de dado, armazenamentos estáticos, sistemas de computação, jogos, realidade virtual, treinamento de redes neurais dentre muitos outros [11].

Um dos serviços disponibilizados também pela AWS é o Elastic Cloud, ou EC2. Nela, é possível dimensionar a capacidade de computação necessária, de forma que elimina a necessidade de investir em um hardware de início. O Amazon EC2 também permite a expansão ou a redução para gerenciar as alterações de requisitos ou picos de popularidade, reduzindo, assim, a sua necessidade de prever o tráfego do servidor.

O serviço realiza a cobrança por tempo de utilização, existindo diferentes variações deste.

Para a realização das análises, foi utilizado uma instância do tipo t2.micro.

3.7 Servidor do IC

O Instituto de Computação disponibiliza para seus alunos uma partição no servidor para que seja possível acessar arquivos remotamente. Essa partição pode ser acessada remotamente por HTTP.

Esse partição foi utilizada para simulação de um cliente, que fazia acesso aos outros serviços de cloud, e realizava upload ou download de arquivos.

3.8 Youtube

O YouTube é uma plataforma de compartilhamento de vídeo, criada por um grupo de pessoas independentes, e depois comprada em 2016 pela Google. Ele se tornou uma das principais ferramentas de distribuição de conteúdos de vídeo no mundo todo e é de uso gratuito para hospedagem de vídeos e para transmissão.

O YouTube foi utilizado como ferramenta auxiliar de armazenamento de vídeo, da qual era possível fazer o download do arquivo de vídeo armazenado, ou streaming do mesmo. Como se trata de um serviço amplamente utilizado no mundo todo, o cenário que utiliza o YouTube para a distribuição de arquivos de vídeo reflete uma situação cotidiana muito

comum. Para realizar o download dos vídeos por essa plataforma, utilizou-se a biblioteca Pytube.

4 Metodologia

O desenvolvimento do projeto consistiu em executar um processamento de vídeos em serviços remotos distintos e em máquinas locais, variando o tipo de arquivo quanto à duração do vídeo e tamanho do conteúdo. Para manter consistência entre os arquivos processados, foi criado um conjunto de onze clipes de vídeos padronizados para ser utilizado em todos os casos de teste, cujas durações variam entre 10 e 600 segundos, todos com a mesma resolução e dimensão.

O processamento foi feito através de um script Python. O script é capaz de abrir um clipe de vídeo que esteja salvo localmente, em diversos formatos, extrair a faixa de áudio do clipe e depois salvar a faixa de áudio separadamente como um novo arquivo no formato MP3. Este script de processamento de vídeo utiliza o pacote Editor da biblioteca MoviePy, que contém a funcionalidade de extrair faixas de áudio e pode ser utilizada de maneira simples.

Um outro script, também em Python, foi criado mas com o intuito de fazer o download de vídeos da plataforma do YouTube. Este script de download utiliza o módulo YouTube da biblioteca pytube, que permite baixar vídeos da plataforma através da URL pública do mesmo. O script de download baixa o clipe de vídeo e salva em um arquivo local. A biblioteca pytube ainda permite fazer o download do vídeo em todos os formatos e resoluções disponíveis na plataforma, mas para trabalhar com o mesmo formato do conjunto de vídeos criados para os testes locais, os vídeos foram baixados em MP4 e na mesma resolução que foram criados.

Os dois scripts, de download e de processamento, nos casos necessários, foram associados de forma a baixar o vídeo do YouTube, gerar um arquivo local temporário com o clipe de vídeo, e depois extrair a faixa de áudio deste clipe de vídeo. Isso é uma das maneiras de fazer de forma transparente a ação de extrair a faixa de áudio de um clipe de vídeo que não esteja salvo na mesma máquina previamente, mas sim em um serviço remoto, como o do YouTube. Para representar máquinas distintas sendo utilizadas remotamente, foram utilizadas duas máquinas remotas: uma provida pelo serviço Google Cloud e outra na Cloud do Instituto de Computação (IC). Dependendo do experimento, os servidores eram os responsáveis por baixar os arquivos de vídeo e/ou processá-los.

Com isso, foram definidos quatro cenários para análise. Cada cenário foi montado considerando casos de uso que representassem situações de aplicações reais, de forma que toda interação começa no lado de um cliente que define qual clipe de vídeo será processado, e ao final o mesmo cliente tem o arquivo da faixa de áudio do vídeo em questão. Os cenários são:

1. O vídeo fonte é armazenado localmente e deseja-se realizar o processamento localmente também;

2. O vídeo fonte é armazenado localmente e deseja-se realizar o processamento no servidor;

3. O vídeo fonte é armazenado em uma plataforma, como o Youtube, e deseja-se que dado uma URL do mesmo, fazer o download do arquivo de vídeo e processar localmente;

4. O vídeo fonte é armazenado em uma plataforma, como o Youtube, e deseja-se processá-lo no servidor;

Dessa forma, seria possível tirar métricas relacionadas com a comunicação entre cliente e servidor (que ficava fisicamente mais próximo, na cloud do IC, e mais distante, na cloud do Google) e ao processamento dos clipes de vídeo. E com isso, tentar identificar qual cenário teria o melhor desempenho, dependendo das condições de rede e do arquivo a ser processado.

Cada cenário é composto de um conjunto de etapas, que envolvem: conseguir o arquivo de vídeo, processá-lo, e depois enviar o arquivo de áudio resultante. Para facilitar a análise de qual das etapas influencia mais no tempo total de processamento, ou no custo de processamento das aplicações, cada uma dessas etapas foram analisadas individualmente e depois, compostas para montar cada um dos cenários. As análises consideradas foram:

- Tempo de processamento dos clipes de vídeo
- Tempo de transferência dos arquivos de vídeo a serem processados
- Tempo de transferência dos arquivos de áudio processados
- Tempo de download dos arquivos de vídeo da plataforma do YouTube

Com isso, é possível descrever como cada cenário foi analisado:

Para o primeiro cenário, temos que vídeo fonte foi armazenado localmente e desejava-se realizar o processamento localmente também. Dessa forma, não há influência da rede nesse cenário nem latências de comunicação. Assim, a análise ficará restrita ao tempo de processamento do vídeo na máquina utilizada. Para este cenário, o único passo foi executar o script de processamento 100 vezes para o conjunto de vídeos padronizado.

Para o segundo cenário, temos que o vídeo fonte está armazenado localmente na máquina cliente, o mesmo é enviado ao servidor, onde deseja-se realizar o processamento. Nesse cenário, todos os vídeos do conjunto padronizado foram disponibilizados em uma máquina e foram adquiridos pelos servidores utilizando-se wget, como descrito na seção 5.1.

Após a transferência dos arquivos de vídeo, executou-se o script de processamento e por fim foi realizada a transferência dos arquivos finais de audio. Para a simplificação desse último passo, foi realizado a transferência no sentido inverso, ou seja, os arquivos de áudio foram transferidos da máquina local para os servidores utilizando wget.

Para o terceiro cenário, onde desejava-se obter um vídeo que é armazenado remotamente na plataforma YouTube e realizar o processamento do mesmo localmente. Foi necessária apenas a realização do download dos arquivos de vídeo dos servidores do YouTube, pois o processamento dos vídeos já havia sido realizado nos cenários anteriores e podiam ser

utilizados para compor esse cenário. Para o download dos vídeos, foi utilizado o script de download de vídeos.

Já para o quarto cenário, desejava-se delegar tanto o download do vídeo do YouTube quanto o processamento para o servidor. Dessa forma, em adição à análise previamente realizada de processamento do vídeo no servidor e da transferência do arquivo de áudio, foi necessária uma análise voltada ao tempo de download do YouTube partindo do servidor. Aqui, também foi utilizado o script de download de vídeos do YouTube.

5 Análise e resultados

5.1 Conjunto padrão de vídeos para experimentação

Para padronizar os resultados e experimentos envolvendo clipes de vídeo, foi criado um conjunto de vídeos para serem utilizados como padrão em todos os experimentos de transferência de dados, bem como nos experimento de processamento dos clipes de vídeo para extração da faixa de áudio.

O conjunto contém 11 clipes de vídeo, onde todos os vídeos têm a mesma resolução (1280 x 720) e variam apenas na duração do clipe de vídeo, e consequentemente a duração da faixa de áudio (entre 10 e 600 segundos).

O conjunto dos vídeos criados pode ser observado na tabela 1, que relaciona a duração do clipe de vídeo com o tamanho necessário para o seu armazenamento.

Duração do clipe de vídeo (segundos)	**Tamanho do arquivo (Mega bytes)**
10	1.0
20	2.5
45	6.5
60	8.4
90	12.0
120	17.6
150	20.9
180	24.8
300	40.5
420	59.4
600	87.2

Tabela 1: Conjunto de vídeos criados para manipulação com valores de duração e tamanho do arquivo

Todos os vídeos marcados na tabela 1 utilizados estavam na resolução de 1280 x 720 e no formato MP4.

A ideia de utilizar o mesmo conjunto de vídeo para todos os testes, é que isso permite comparar os resultados entre experimentos distintos mais facilmente.

O mesmo conjunto de vídeos também foi enviado para a plataforma do YouTube para que pudessem ser utilizados quando necessário.

5.2 Desempenho do processamento local de clipes de vídeo

Para compreender melhor o comportamento da extração da faixa de áudio de clipes de vídeo, e como esse comportamento varia para diferentes clipes de vídeo, fizemos uma análise em um sistema local, ou seja, mantendo todo o processamento em uma única máquina.

O primeiro cenário utilizado para essa análise foram os 11 clipes de vídeo criados como padrão, descritos na tabela 1. Isso permite compreender como o tempo de processamento varia de acordo com a duração do vídeo que está sendo processado.

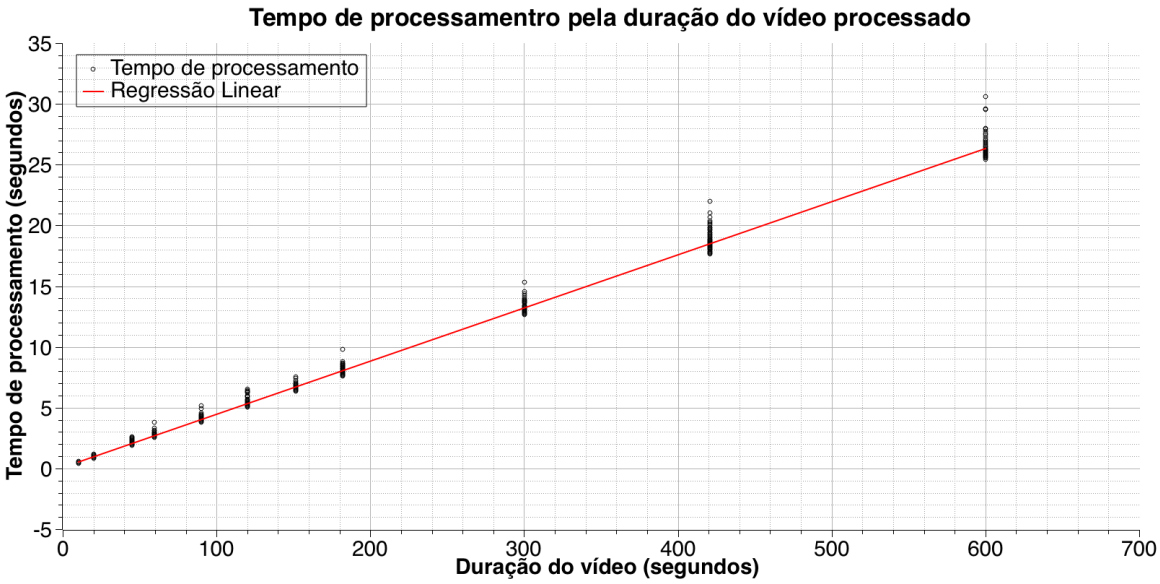

Figura 1: Resultado para o processamento de extração da faixa de áudio para diferentes tamanhos de vídeo, com o processamento feito localmente, ambos os eixos estão marcados em segundos. A máquina utilizada foi um Macbook com um processador de $2,2$ GHz e 6-Core com 16GB de memória RAM. A reta marcada no gráfico foi calculada através de uma regressão linear e tem formato $y = Ax + B$, onde $A = 0.043 \pm 0.000$ e $B = 0.075 \pm 0.018$.

É possível notar na figura 1 que existe uma relação linear entre o tamanho do clipe de vídeo e o tempo de processamento necessário para extrair a faixa de áudio do mesmo. Dessa forma, vídeos maiores demoram mais a ser processados.

Depois, um segundo cenário (cenário 2) foi realizado, dessa vez utilizando o mesmo vídeo mas variando as resoluções para cada clipe. Foram utilizadas 4 resoluções de vídeo (4K, 1080p, 720p e 480p) e para cada clipe executado 100 vezes. Os resultados podem ser observados na figura 2.

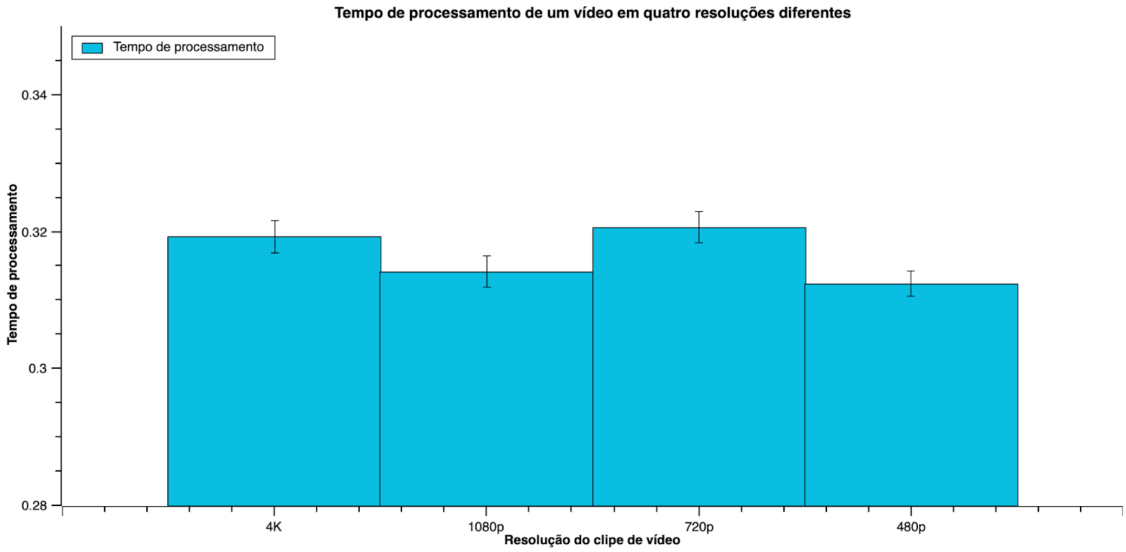

Figura 2: Gráfico do tempo de processamento, em segundos, de um mesmo clipe de vídeo com resoluções diferentes. O vídeo utilizado tinha duração de 4.68 segundos.

Através da figura 2 é possível notar que a resolução do vídeo teve pouco impacto no tempo de processamento total da extração da faixa de áudio. Os valores são muito próximos, e as faixas de erro tem intersecções nos 4 casos analisados, o que indica que o tempo de processamento independe da resolução do arquivo de vídeo, mas sim da duração do clipe sendo processado.

Mesmo que a resolução do clipe de vídeo não tenha dado alterações no tempo de processamento total para extração da faixa de áudio, ela aumenta consideravelmente o tamanho do arquivo de vídeo que está sendo processado, o que implica em uma necessidade maior de memória para execução e armazenamento do vídeo. Os tamanhos dos arquivos de vídeo utilizados podem ser observados na Tabela 2.

Resolução do Vídeo	**Tamanho do arquivo (Mega bytes)**
4K	24.2
1080p	10.2
720p	7.2
480p	2.6

Tabela 2: Tamanho dos arquivos de vídeo utilizados para o cenário de resoluções diferentes

É importante ter analisado esse fator localmente, pois ao trabalhar com sistemas distribuídos, onde queremos enviar o arquivo a ser processado para uma máquina diferente, o tamanho dos arquivos a serem transferidos tem uma grande influência no tempo de comunicação.

5.3 Comparação do processamento local em máquinas distintas

Utilizando o conjunto de arquivos de vídeo descritos na tabela 1, o mesmo experimento do tempo de processamento para extração da faixa de áudio foi utilizado nos serviços do Cloud do IC e do Google Cloud.

Como as máquinas têm especificações diferentes, é esperado que o tempo de processamento em cada uma também seja diferente. Dessa forma, para cada serviço de cloud utilizado neste estudo, foi feito o mesmo experimento de processamento local.

É possível ver os resultados do processamento da Cloud do Google na figura 3 e da Cloud do IC na figura 4.

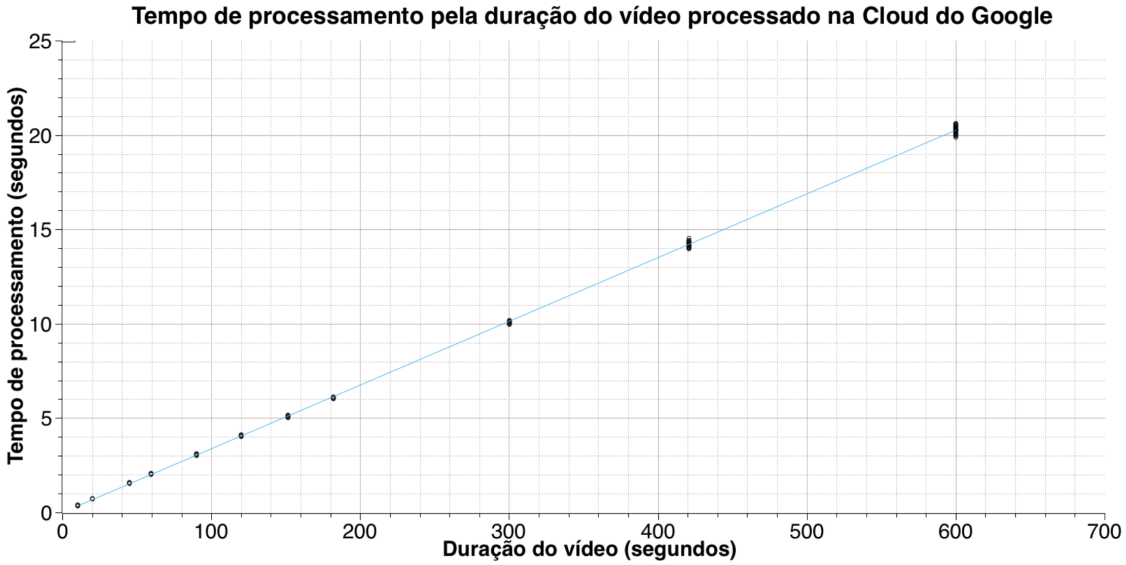

Figura 3: Resultado para o processamento de extração da faixa de áudio para diferentes tamanhos de vídeo, com o processamento feito na Google Cloud, ambos os eixos estão marcados em segundos. A reta marcada no gráfico foi calculada através de uma regressão linear e tem formato $y = Ax + B$, onde $A = 0.033 \pm 0.000$ e $B = 0.004 \pm 0.003$.

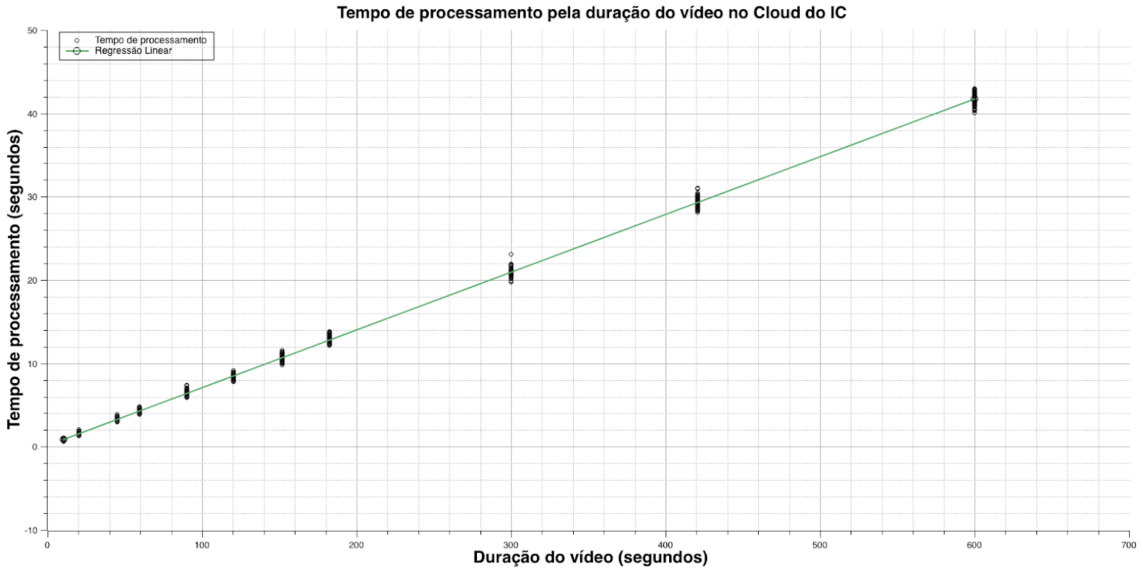

Figura 4: Figura 7 - Resultado para o processamento de extração da faixa de áudio para diferentes tamanhos de vídeo, com o processamento feito na Cloud do IC, ambos os eixos estão marcados em segundos. A reta marcada no gráfico foi calculada através de uma regressão linear e tem formato $y = Ax + B$, onde $A = 0.069 \pm 0.000$ e $B = 0.130 \pm 0.016$.

O comportamento observado tanto nas clouds quando na máquina local, figura 1, são semelhantes, todos tem um crescimento linear com a duração do arquivo de vídeo processado. Mas é possível notar que a reta traçada em cada máquina possui uma inclinação diferente.

A figura 5 mostra um comparativo das 3 curvas observadas. Com isso fica claro que o processo de extração da faixa de áudio de clipes de vídeo é um processamento estável, mas depende da máquina a qual está realizando o processamento.

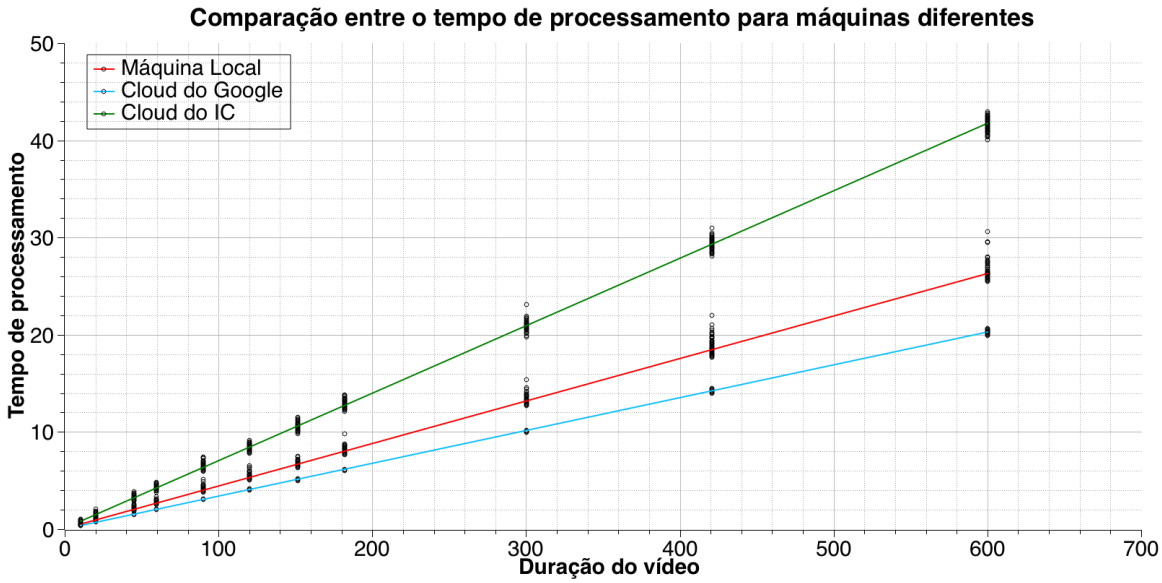

Figura 5: Comparação dos resultados do processamento de extração da faixa de áudio de conjuntos de vídeo para 3 máquinas distintas.

Como o processamento do arquivo de vídeo, apenas, não depende de comunicação com outras máquinas, ele independe de processos de rede que podem ser mais voláteis e sujeitos a uma maior quantidade de erros.

Um ponto interessante na figura 5 é que o tempo de processamento na Cloud do Google foi o que apresentou o melhor resultado. Conforme os arquivos de vídeo processados ficam maiores, a diferença entre o Cloud do Google e as outras máquinas utilizadas também aumenta, de forma que pode ser mais proveitoso ou mais eficiente realizar esse processo na cloud.

5.4 Transferência de arquivos de vídeos usando Wget

Buscando simular o cenário em que o cliente envia um arquivo armazenado em sua máquina para que o servidor possa realizar a extração da faixa áudio do mesmo, utilizou-se o pacote Wget [sessão Conceitos 2.3] para realizar o download de vídeos de diferentes tamanhos, sendo que estes estão armazenados em uma pasta pública do Instituto de Computação.

Para coleta dos tempos, o seguinte contexto foi utilizado: o conjunto de vídeos descrito na tabela 1 foi armazenado no servidor do Instituto de Computação (IC) e os arquivos podiam ser acessados externamente usando HTTP. Para cada arquivo de vídeo, foram transferidos 100 vezes entre o servidor que armazenava os arquivos e o serviço de cloud desejado utilizando o comando Wget para medir a latência de comunicação.

O cenário foi aplicado no serviço de Cloud do IC e os resultados podem ser observados na figura 6.

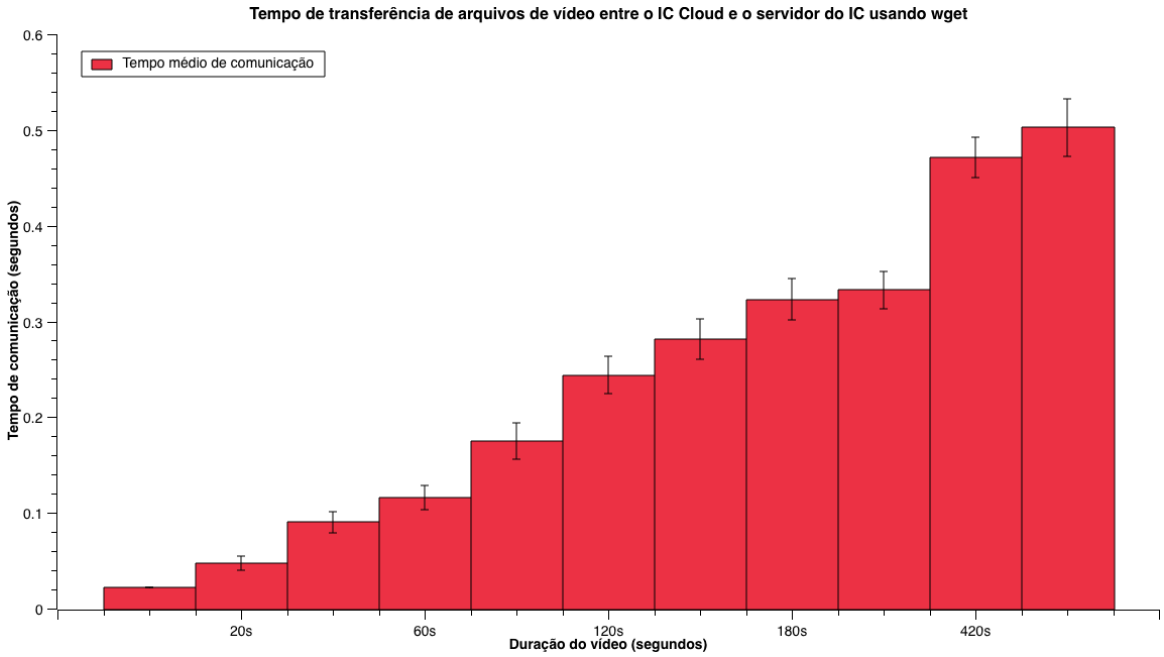

Figura 6: Gráfico do tempo médio de comunicação para transferência de arquivos de vídeo entre o IC Cloud e o servidor do IC usando wget.

É possível notar que o tempo de comunicação aumenta de acordo com o tamanho do vídeo sendo transferido, dado que o tamanho do arquivo que está sendo transferido também aumenta. Era esperado que o tempo de comunicação aumentasse de acordo com o tamanho do arquivo sendo transferido.

Depois dos testes no Cloud do IC, os mesmos testes foram feitos utilizando o mesmo cenário no Google Cloud e tirou-se as mesmas medições de tempo representadas na figura 7.

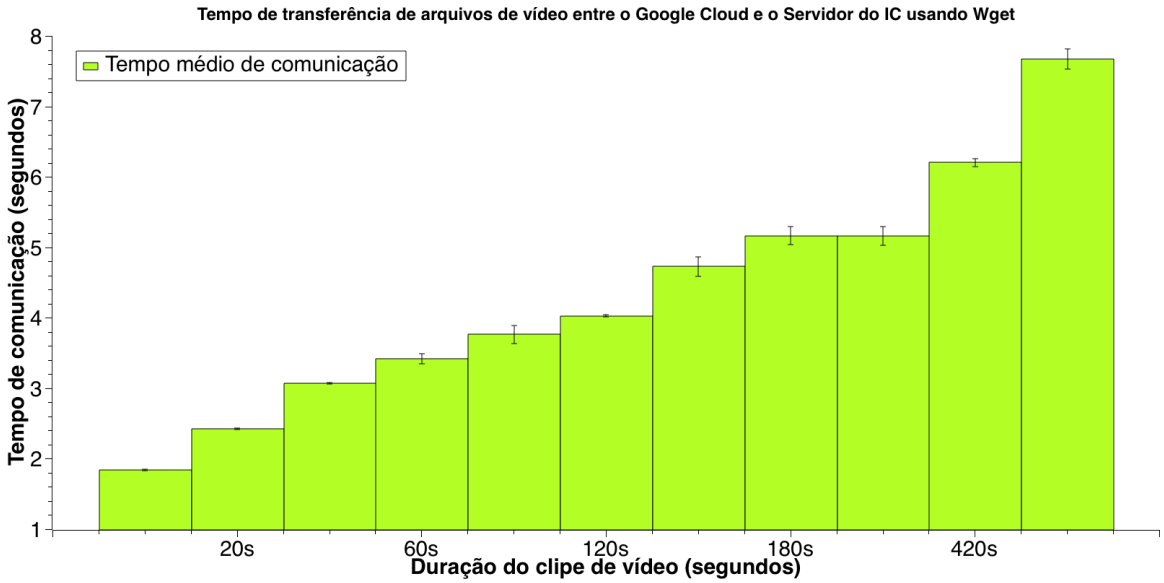

Figura 7: Gráfico do tempo médio de comunicação para transferência de arquivos de vídeo entre o Google Cloud e o servidor do IC usando wget.

Analisando o gráfico, nota-se que existe uma grande diferença entre realizar o download no Cloud da Google e no Cloud do IC. Tomando como exemplo o clipe de vídeo com duração de 420 segundos, o tempo de download para o primeiro foi de 6,2 segundos, enquanto para o segundo tem-se 0,47 segundos.

Apesar do comportamento ser semelhante para as duas máquinas, onde o tempo de comunicação cresce com a duração do clipe de vídeo sendo transferido, os tempos totais de transferência foram muito mais altos no segundo caso para os mesmos arquivos de vídeo.

Como se trata de uma operação de transferência de arquivos pela internet, usando HTTP, a distância geográfica entre os elementos têm grande impacto na latência da comunicação, de forma que quão mais longe estiverem as máquinas se comunicando, maior será o tempo de comunicação.

Além disso, a qualidade da rede, da conexão à internet, das duas máquinas envolvidas na transferência também influencia muito no tempo total de comunicação.

Comparando os gráficos da figura 6 e da figura 7 é possível observar o quanto o parâmetro geográfico influenciou no tempo de transferência dos arquivos de vídeo, dado que a máquina do Google Cloud está localizada na Carolina do Norte, nos Estados Unidos, enquanto a máquina da Cloud do IC está muito próxima da onde a informação está armazenada (também no IC).

Pode-se atribuir a grande diferença entre o tempo de comunicação entre os servidores o fato de que a rede tem uma influência muito maior nos testes relacionados ao Google Cloud. Dessa forma, tem-se mais variantes que podem vir a afetar esse cálculo.

5.5 Transferência de arquivos de áudio usando Wget

Para simular a resposta das máquinas remotas ao cliente, realizaram-se medições acerca do download do áudio gerado pela extração de áudio do vídeo, sendo que essa rotina foi estruturada da mesma forma do item anterior: utilizou-se o pacote Wget [sessão Conceitos 2.3] para tal.

Visando a coleta dos tempos, com o mesmo conjunto de vídeos sendo usados nos experimentos, após utilizar o script de extração do áudio dos vídeos, o resultado - no caso, os áudios extraídos - ficaram armazenados no Instituto de Computação (IC). Assim era possível acessá-los externamente usando HTTP.

Cada arquivo de áudio foi transferido 100 vezes entre o servidor do Google Cloud e do Cloud do IC utilizando o comando Wget para medir a latência de comunicação.

Com os dados obtidos, estruturou-se o gráfico da Figura 5.

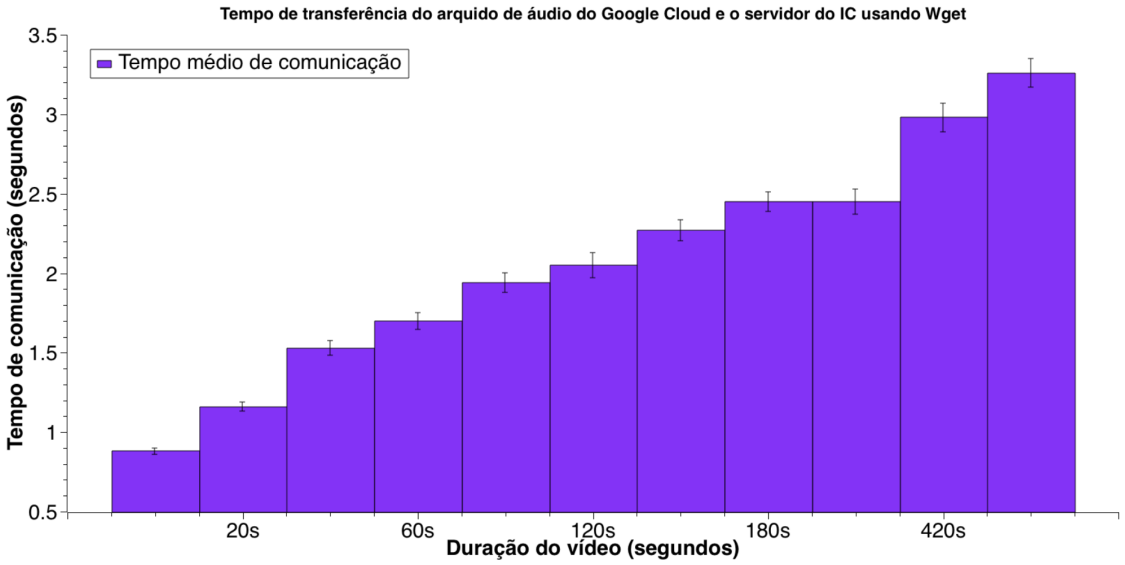

Figura 8: Gráfico do tempo médio de comunicação para transferência de arquivos de áudio entre o Google Cloud e o servidor do IC usando Wget.

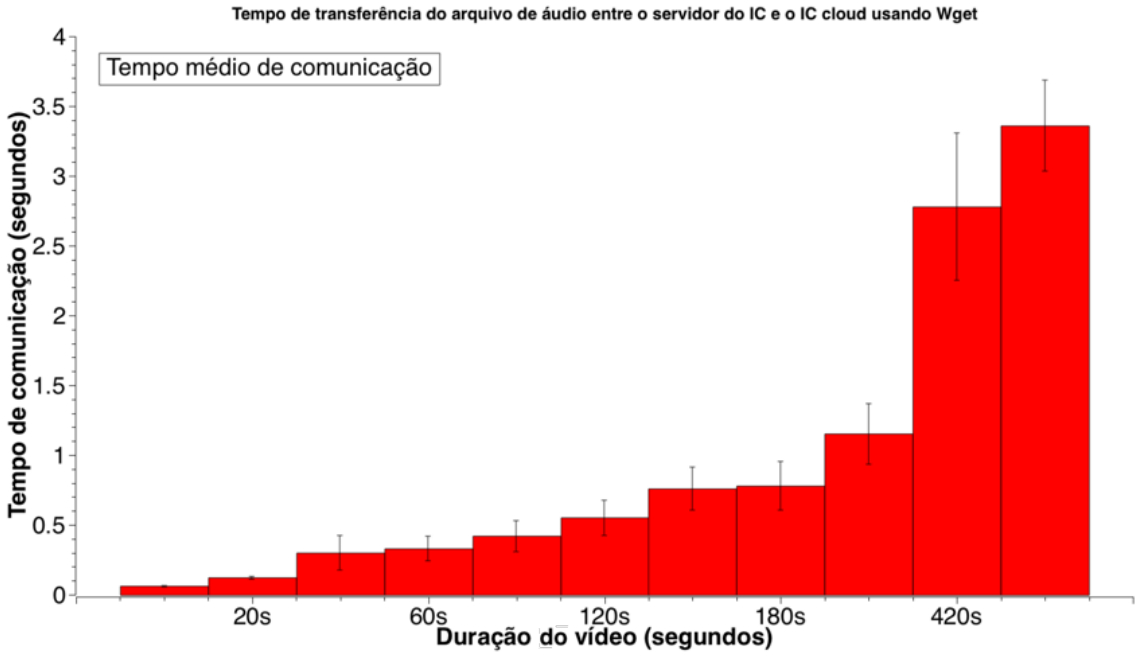

Figura 9: Gráfico do tempo médio de comunicação para transferência de arquivos de áudio entre o IC Cloud e o servidor do IC usando Wget.

Comparando os gráficos 8 e 9, é possível concluir que existe um comportamento similar à seção anterior, isto é, a transferência de arquivos foi influenciada diretamente pelas questões geográficas - o tempo gasto no caso do Google Cloud é maior do que o do caso do IC Cloud, o que se justifica pela maior proximidade do IC Cloud com o servidor de armazenamento que usamos no IC - e por questões de rede também.

A rede pode ter influenciado principalmente para os arquivos maiores, como o caso do áudio de 600s, em que a constância na transferência pode variar e com isso o tempo dispensado nesta também.

5.6 Transferência de arquivos de vídeo do Youtube

Para explorar o cenário em que o vídeo encontra-se distribuído em um serviço de terceiros, simulou-se o download de vídeos armazenados na plataforma do Youtube, descrito na seção 3.8.

Partindo das instâncias, respectivamente Google Cloud e IC Cloud, utilizou-se a API do Pytube, descrita na seção 3.3, para realizar o download.

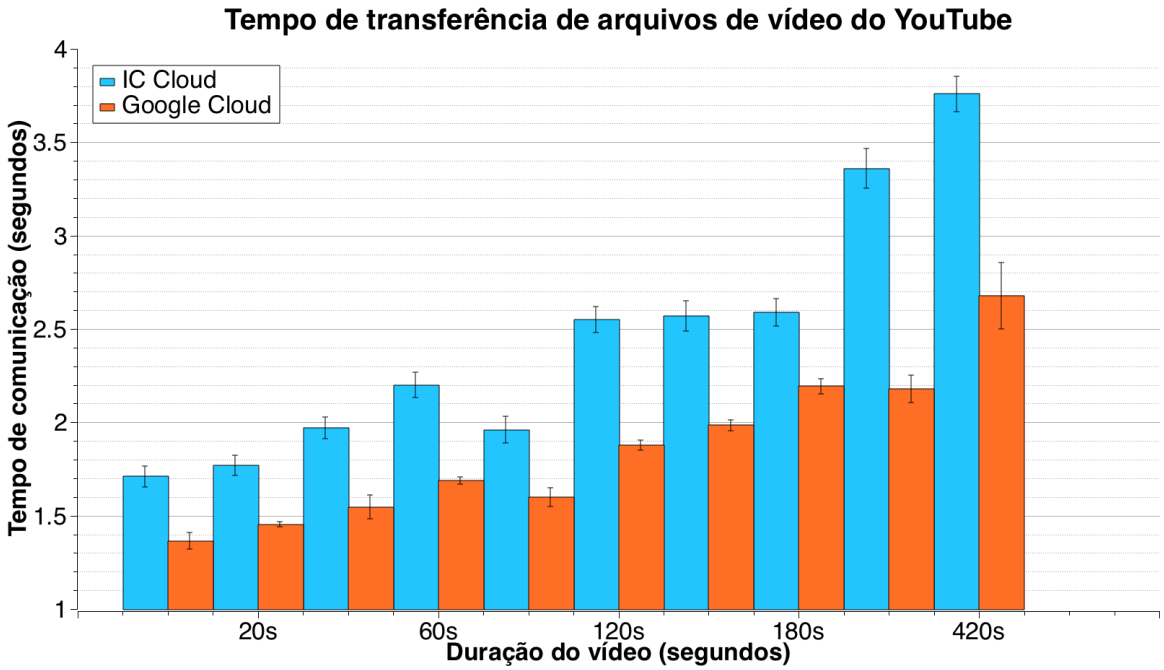

Figura 10: Comparação entre os tempos médios de download de arquivos de vídeo da plataforma do YouTube no Cloud do IC e no Google Cloud.

Analisando o gráfico 10, é possível analisar que os tempos de download partindo do Google Cloud tem tempo menor do que no caso do IC Cloud. Isso pode ser diretamente relacionado à localização geográfica: segundo os dados mais recentes do Google, seus centros de armazenamento se concentram principalmente nos Estados Unidos e na Europa [google distributions]. Dessa forma, por estar localizado na Carolina do Norte, o Google Cloud tem vantagem nessa comparação.

Além da questão física da distância, existem fatores como as chances de ocorrer interferências são maiores e depende-se bem mais da qualidade da rede envolvida.

6 Cenários de estudo

6.1 Cenário 1 - Processar o vídeo localmente

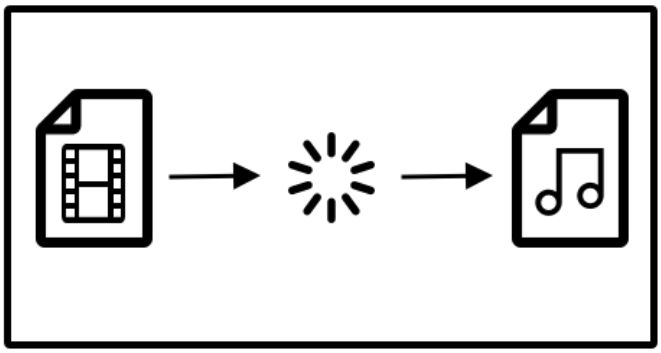

Figura 11: Esquema do cenário 1, um arquivo de vídeo (à esquerda) é processado dentro da mesma máquina (ao centro) e gera um arquivo de áudio (à direita). Os limites externos indicam que todo o processamento e arquivos estão dentro da mesma máquina, bem como os arquivos de áudio e de vídeo.

Este cenário depende apenas do processamento do arquivo de vídeo feito localmente, então não está sujeito à problemas de rede, nem a condições de rede para download de arquivos pois não depende de fazer comunicações com outras máquinas.

Mas existem pontos de atenção nesse cenário que são:

- O cliente deve ter o vídeo a ser processado previamente, fazendo um gasto maior de memória de armazenamento;
- O cliente também deve ter as bibliotecas e scripts capazes de realizar esse processamento;

Não é um cenário ideal se considerarmos dispositivos que tenham pouca capacidade de armazenamento ou de processamento e que busquem apenas do resultado final, ou seja, o arquivo de áudio.

Além disso, nesse cenário, a máquina cliente deve ter as bibliotecas necessárias, e deve ser capaz de processar o arquivo de vídeo, estando sujeito às condições variáveis da máquina em questão e possíveis incompatibilidades de versão. Portanto, além do tempo adicionado, deve ser levada em consideração a necessidade da configuração do ambiente.

Analisando os dados (Tabela X em anexo) coletados, tem-se que o único passo a ser medido será o tempo de processamento local, visto que os vídeos a serem processados e os

áudios gerados pelo processamento deverão permanecer na máquina (não há necessidade de envio de dados). Dessa forma, o único fator que influencia diretamente nesse cenário seria a configuração do ambiente em que o vídeo é processado.

6.2 Cenário 2 - Processar remotamente um arquivo salvo

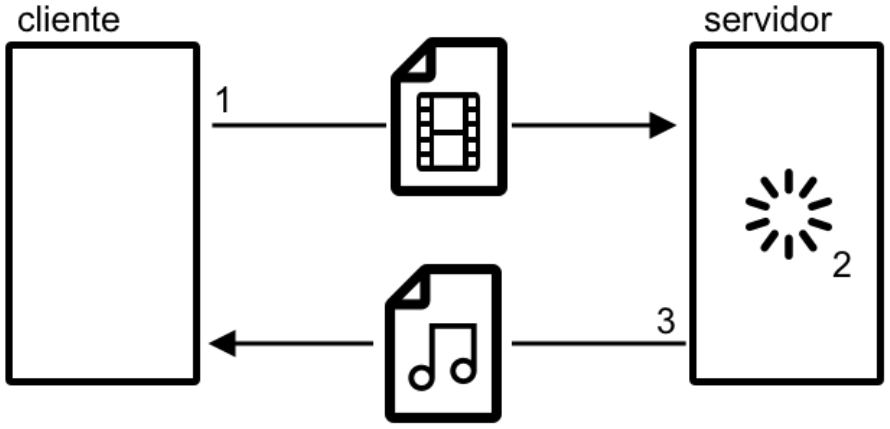

Figura 12: Esquema do cenário 2, uma máquina cliente envia um arquivo de vídeo para uma máquina servidor (1), onde é processado (2), gera um arquivo de áudio que é enviado ao cliente ao final (3).

A diferença entre o cenário 2 e o cenário 1 é que o processamento é feito agora em uma máquina remotamente, mantendo a mesma estrutura anterior de armazenamento de dados. Com isso, um novo fator é inserido dentro do sistema que é a necessidade de realizar uma transferência de arquivo via internet. Dessa forma, neste cenário, temos influência da rede, da conexão entre as máquinas, tanto para enviar o arquivo de vídeo a ser processado, quanto para receber o arquivo de áudio resultado do processamento.

Como esse novo fator de comunicação é adicionado, existe um novo fator de latência que afeta o tempo total para o processamento do arquivo de vídeo. O tempo de transferência do arquivo de vídeo pode ser melhor observado na figura 7, e o tempo de transferência do arquivo de áudio na figura 8.

O tempo total de comunicação pode ser descrito então como a soma do tempo de envio do vídeo, mais o tempo de envio do arquivo de áudio. Se tomarmos o servidor do Google Cloud como base, que é o servidor mais distante dos dados testados, os tempos de transferência podem ser observados nas figuras 4 e 6. Pode-se notar que o tempo de transferência, para algumas instâncias de vídeo, varia entre 2 a 10 segundos de comunicação.

Um outro ponto importante, é que o processamento do vídeo, representado na etapa 3, não está sendo mais realizado na mesma máquina que no cenário 1, o que traz um novo fator de mudança para o tempo de processamento do cenário como um todo. A comparação

entre os tempos de processamento das máquinas utilizadas pode ser observada na figura 7.

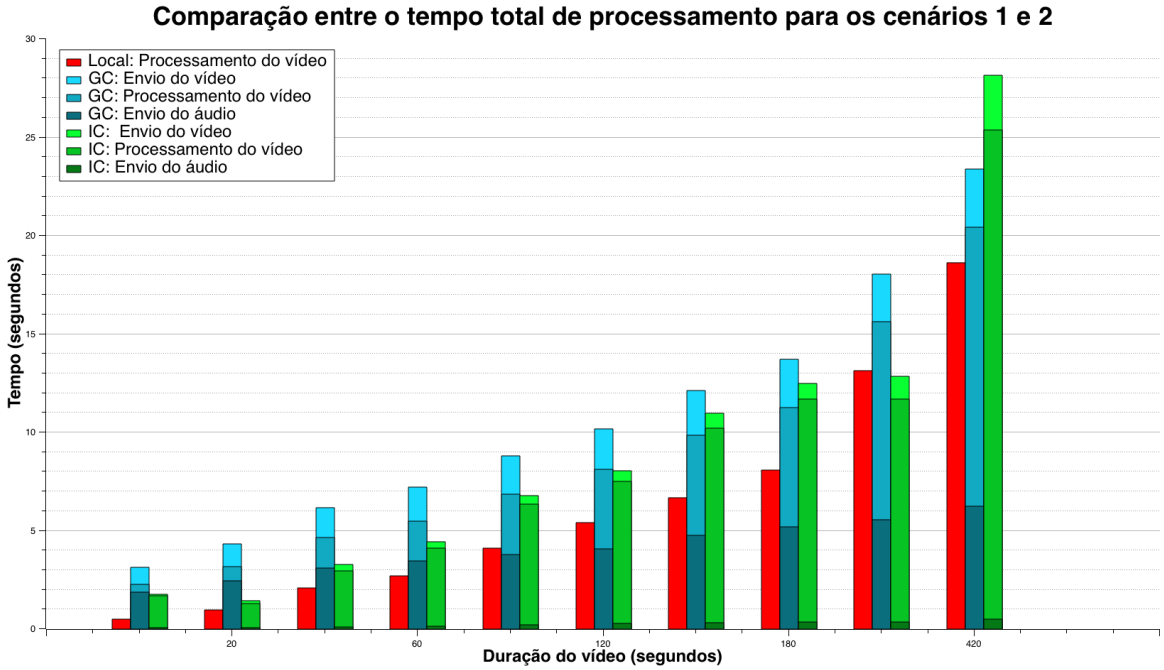

Figura 13: Comparação entre o tempo total de processamento entre os cenários 1 (colunas da esquerda) e 2 (colunas da direita). Os tempos estão discriminados dentro das barras em relação à etapa de processamento envolvida. GC: Google Cloud, IC: IC Cloud.

A figura 13 mostra uma comparação entre os tempos totais de processamento entre os cenários 1 e 2 para as máquinas do Google Cloud e do IC Cloud.

Como discutido anteriormente, o tempo total de processamento é maior no cenário 2 para as duas máquinas, em relação ao cenário 1.

No caso do Google Cloud, a faixa referente à extração da faixa de áudio do clipe de vídeo (processamento) é mais eficiente que da máquina utilizada como referência de processamento local. Conforme o tamanho dos vídeos cresce, a latência de comunicação passa a ficar cada vez menor em relação ao tamanho total do vídeo, fazendo com que o processamento no servidor se torne mais vantajoso até mesmo em questão do tempo total de processamento.

Um outro ponto que pode ser comparado é da latência de transferência do arquivo em relação à distância do servidor. Se tomar apenas os tempos de comunicação, de transferências de arquivos, o Cloud do IC teve um desempenho bem melhor que o Google Cloud na maioria dos casos. Isso evidencia que ao colocar parte do serviço sendo realizado em uma máquina remota, a distância dessa máquina pode influenciar muito na latência de comunicação, e quão mais próximo do cliente, menor será essa latência. Todavia, existem mais fatores envolvidos, no caso a máquina utilizada no IC Cloud tinha um poder de processamento pior em relação à do Google Cloud, de forma que para vídeos maiores, o tempo total de processamento no Google é mais rápido que no cloud do IC, mesmo com um tempo de comunicação maior.

Isso mostra que o cenário ideal da arquitetura cliente-servidor proposta, depende se vários fatores. É importante analisar tanto proximidade entre as máquinas envolvidas no processo, bem como a capacidade de processamento. Todavia, máquina com processamentos melhores tem custo maior de operação, então também é um fator a ser considerado para definir o cenário ideal.

6.3 Cenário 3 - Processar localmente um vídeo do YouTube

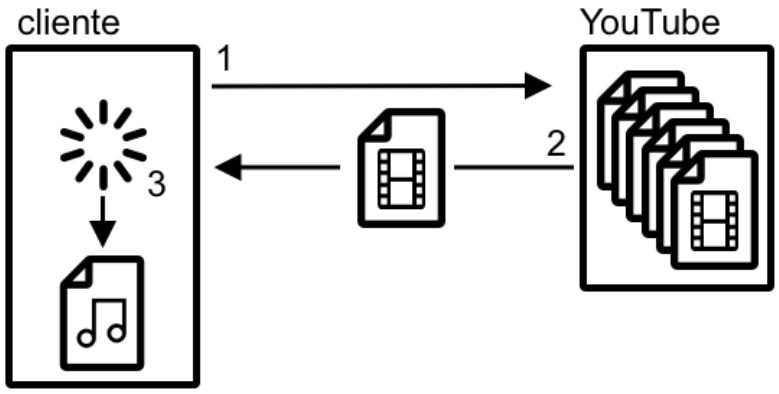

Figura 14: Esquema do cenário 3, uma máquina cliente pede um vídeo para o YouTube (1), que lhe é enviado pela plataforma (2) e depois processado localmente (3). Após o processamento o arquivo de áudio já está armazenado na máquina desejada.

Este cenário se difere do primeiro por não ter o arquivo de vídeo salvo localmente, e sim em um serviço externo que permite que o arquivo de vídeo seja recuperado, no caso, o YouTube, dessa forma, este cenário está sujeito às condições de rede para fazer o download do arquivo da plataforma.

Além da latência de comunicação, em relação ao cenário 1, este cenário introduz uma mudança na estrutura do processamento que permite melhorar a escalabilidade do sistema. Em comparação com os cenários 1 e 2, onde o arquivo era armazenado previamente na máquina cliente, aqui no cenário 3 ele está num serviço externo. Tratando-se de uma aplicação real, esse é o primeiro cenário tratado em que o cliente não precisaria ter obtido o vídeo previamente, nem precisa gastar memória para manter esses arquivos armazenados.

Os tempos da etapa 3 de processamento do vídeo são os mesmos do cenário 1, por se tratar da mesma máquina e da mesma computação.

Adicionando o tempo de comunicação (download do vídeo da plataforma do YouTube) é possível obter o tempo total para obter o arquivo de áudio.

Como foi inserido o fator da latência de comunicação e transferência do arquivo de vídeo, os tempos totais de processamento do cenário 3 são maiores que os do cenário 1,

invariavelmente. Mas essa diferença de tempo está diretamente associado ao ganho de não ser mais necessário armazenar o arquivo de vídeo localmente e poder tirar proveito de uma plataforma especializada nisso.

Os dados coletados para esse cenário podem ser visualizados na Tabela [cenario-tres].

6.4 Cenário 4 - Processar remotamente um vídeo do YouTube

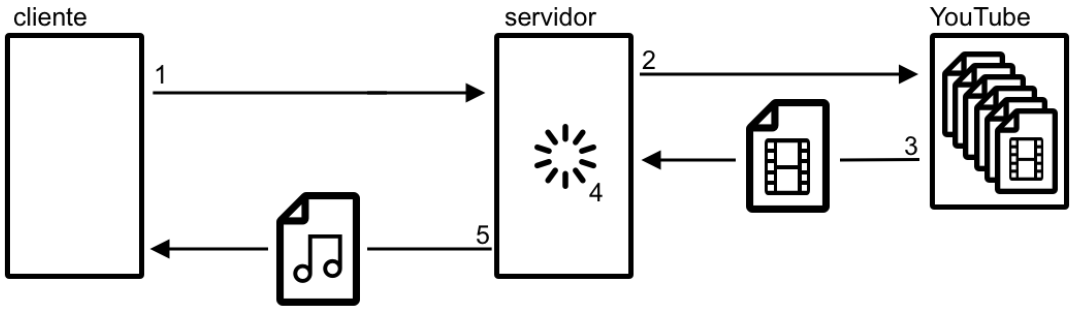

Figura 15: Esquema do cenário 4, uma máquina cliente pede um arquivo de áudio para o servidor enviando a referência (URL) do arquivo para ser processado (1). O servidor então busca o arquivo de vídeo necessário no YouTube (2), que envia um arquivo de vídeo pro servidor (3), que então é processado (4) e envia um arquivo de áudio de volta para o cliente (5).

Este é o cenário mais simples do lado cliente, e também o cenário que agrega os pontos positivos criados nos cenários 2 e 3 em relação ao cenário 1, onde o processamento é delegado para uma máquina remota e a responsabilidade de armazenamento dos arquivos de vídeo para um serviço externo, o YouTube.

Em contrapartida, é o cenário mais complexo e mais exigente do lado do servidor, colocando-o para fazer download e processamento do arquivo, e depois envio do arquivo de áudio resultante.

Existem três momentos que majoram o tempo necessário para a execução total desse fluxo, como descritos na figura 15, o download do arquivo de vídeo do YouTube em 3, o processamento do vídeo em 4, e o envio do arquivo de áudio em 5. As etapas 1 e 2 têm tempos de duração que podem ser desprezados em relação às outras, pois tratam apenas de uma única requisição HTTP, sem transferir nenhum tipo de arquivo, que faz com que sejam unitárias e muito mais rápidas que as demais requisições.

É possível observar um comparativo do tempo total de processamento, ou seja, até que a máquina cliente tenha o arquivo de áudio final, entre todos os cenários na figura 16.

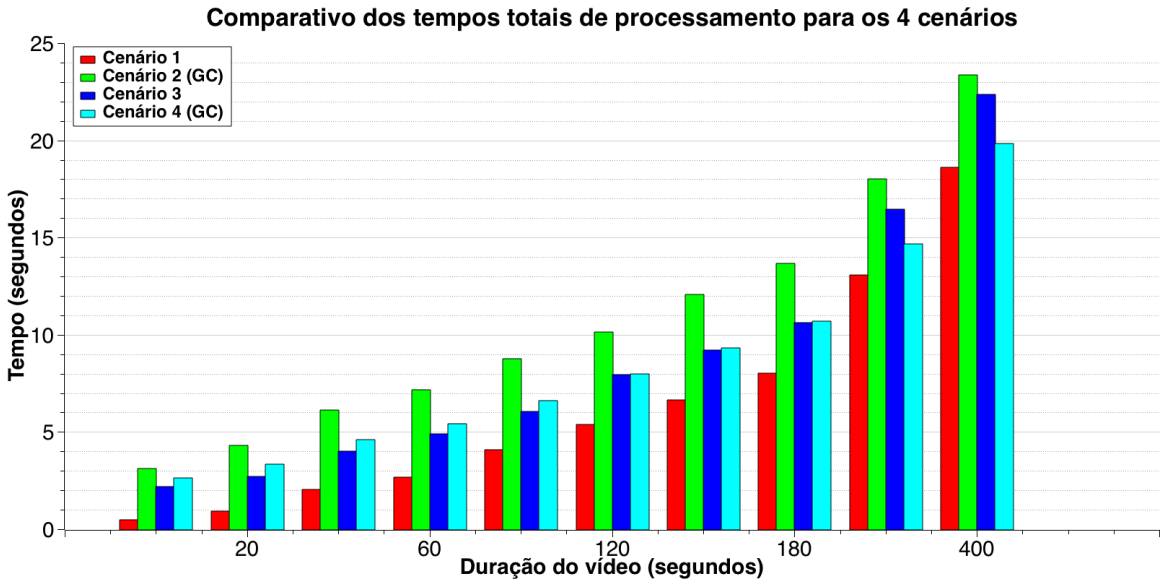

Figura 16: Comparação dos tempos totais de processamento entre os 4 cenários montados. O servidor utilizado para os cenários 2 e 4 foi o Google Cloud (GC), que obteve resultados melhores que o Cloud do IC.

Cada cenário tem um comportamento semelhante que é de aumentar gradativamente o tempo total de processamento com o aumento do tamanho do arquivo processado. Mas a maneira como os diferentes cenários crescem ao longo do tempo que deve ser analisada.

Uma relação interessante é uma comparação dos cenários 1 e 4. Como é possível ver na figura 5, o servidor do Google Cloud processa em menor tempo os arquivos de vídeo que a máquina utilizada para os testes locais. Para arquivos de vídeo muito grandes, essa diferença também aumenta. Agora, no gráfico 16 é possível notar que a diferença entre os cenários 1 e 4 diminuem gradualmente conforme o tamanho do arquivo processado aumenta.

Dessa forma, é possível prever uma situação onde o cenário 4 seria mais vantajoso que o cenário 1, mesmo considerando o tempo de comunicação e transferência dos arquivos. Mas isso decorre que a máquina utilizada no servidor, teve um desempenho melhor na etapa de processamento que a máquina utilizada localmente. Se isso não fosse verdade, esse cenário jamais seria vantajoso.

Como o servidor utilizado foi de um serviço terceiro, é possível que uma máquina com capacidade de processamento melhor seja alocada, melhorando mais ainda os resultados, mas aumentando o custo de uso do serviço.

Uma outra análise importante a partir da figura 16 é que o cenário 4 passa a ter um resultado melhor, ou seja, um tempo total de processamento menor que o cenário 3 conforme o tamanho dos arquivos processados aumenta. Isso se deve ao fato de que o tempo de download do arquivo do YouTube foi menor que o envio do arquivo localmente para o servidor. Isso evidencia que o local onde os arquivos serem processados estão armazenados também influencia em muito o tempo total de processamento quando existe comunicação envolvida.

7 Lições aprendidas

Inicialmente, almejava-se realizar a identificação de objetos em um vídeo. Dado um link da plataforma de vídeos Youtube, utilizava-se a biblioteca chamada Pytube (seção 3.3) que fazia o download do vídeo e realizava-se o processamento com o auxílio do TensorFlow. Para tal, alocou-se duas máquinas para realizar esse processamento remoto: uma no Google Cloud (seção 3.4) e outra no Cloud do IC (seção 3.5).

Como a análise dependia da comparação entre o processamento do vídeo local e o processamento na nuvem, implementou-se um servidor flask, descrito na seção 7.1.2, que, ao receber uma requisição com um link do YouTube, baixava o vídeo e utilizava um script em python para processá-lo. E a fim de facilitar a reutilização dos serviços e para garantir o mesmo processamento entre todas as máquinas, foi utilizado Docker para agrupar todos os serviços em uma imagem (servidor flask, nginx para o proxy reverso e script python).

Entretanto, a rede neural utilizada era muito pesada e requisitava muita memória, o que impossibilitou execuções em volume (que seria um dos cenários de teste a serem explorados). Além disso, a rede neural demorava consideravelmente para processar um vídeo, excedendo o tempo de timeout da requisição para vídeos muito longos. Dessa forma, o conjunto de vídeos que poderiam ser utilizados como base de análise ficou muito restrito. Em adendo, as imagens geradas pelo Docker, apesar de facilitarem no deploy nas máquinas a garantirem o ambiente, ocupavam muita memória, inviabilizando o uso de instâncias menores, como os do AWS EC2 (seção 3.6).

Devido a todos esses empecilhos, decidiu-se quebrar as análises de tempo, medindo-se cada parte separadamente, bem como trocar esse processamento de identificação de objetos em vídeo por extração de áudio do vídeo.

7.1 Tecnologias adicionais estudadas

7.1.1 Tensorflow

TensorFlow é uma biblioteca de código aberto para criação de aplicações e modelos em Machine Learning, foi originalmente desenvolvida pela Google Brain Team na organização de pesquisa Machine Intelligence do Google para aprendizado de máquina e pesquisa de redes neurais profundas [13].

A biblioteca também é cross-plataforma e é capaz de rodar em uma grande quantidade de plataformas, CPUs e CPUs, bem como em hardwares mobile e sistemas embarcados [14].

Sua importância no projeto foi uma das bibliotecas utilizadas para carregar os modelos de Machine Learning utilizados para a detecção de imagens em vídeos.

7.1.2 Flask

Flask é um framework WSGI (em português, Interface de Porta de Entrada do Servidor Web) leve de aplicações web. Ele foi feito para ser simples de utilizar com a habilidade de escalar aplicações complexas. Ele começou como uma camada em cima de outros frameworks WSGI (Werkzeug e Jinja), mas acabou se tornando um aod frameworks mais populares de aplicações web utilizando Python [10].

7.1.3 Docker

Docker é uma ferramenta que ajuda na criação de aplicações utilizando containers, que permitem que o desenvolvedor empacote sua aplicação junto com todas as dependências, garantindo que o conjunto seja capaz de ser executado em qualquer máquina da mesma forma.

O Docker foi utilizado no projeto para ajudar no *deploy* e a rodar a aplicação em todas as instâncias utilizadas (cloud fornecida pela Google, cloud do IC)

7.1.4 ImageAI

ImageAI é uma biblioteca de código aberto, criada por Moses Olafenwa e John Olafenwa, irmãos, e desenvolvida em Python que permite desenvolver aplicações e sistemas que que contém capacidades de Visão Computacional e Deep Learning de maneira simples. A biblioteca suporta uma grande quantidade de algoritmos de

Machine Learning para predição de imagens, detecção de objetos em imagens e vídeos, e também mecanismos para treinamento de redes para detecção e previsão de objetos customizados. A parte de detecção de objetos em vídeos e imagens usa as redes RetinaNet, YOLOv3 (do inglês, você vê apenas uma vez) e a TinyYOLOv3 treinadas com o COCO dataset [16].

ImageAI foi a biblioteca que proveu o modelo de Machine Learning utilizado inicialmente no projeto para detecção de imagens, a YOLOv3, treinada no COCO dataset.

7.1.5 YOLO

YOLO - You Only Look Once (Você vê apenas uma vez), é um sistema de detecção de objetos em tempo real criado na Universidade de Washington [15].

A rede YOLOv3, que é a terceira versão da rede YOLO, foi o modelo utilizado neste projeto inicialmente como o algoritmo de detecção de objetos em imagens e vídeos. Mas o uso da rede foi descartado quendo o escopo de estudo foi alterado.

8 Conclusão

O intuito deste projeto foi averiguar o comportamento de variações de infraestrutura em Fog para o processamento de arquivos de vídeo, buscando cenários mais vantajosos em termos do tempo de resposta, do custo de alocação e da qualidade do serviço oferecido considerando situações onde o serviço é consumido por usuários que não tenham o poder de processamento em seus dispositivos locais, como celulares, tablets ou até mesmo notebooks com processadores mais simples.

Uma das dificuldades encontradas foi a de encontrar um ambiente em Fog que pudesse atender as demandas de depuração - nosso servidor em Cloud do IC não tinha o hardware disponível para executar algoritmos de Aprendizado de Máquina que utilizavam o framework do TensorFlow, logo outra funcionalidade deveria ser utilizada como métrica, chegando-se assim na análise da extração das faixas de áudio de clipes de vídeo.

Considerando os tempos de processamento de instância envolvida no estudo, fica evidente que a capacidade de processamento da máquina em questão pode ter grande influencia nos tempos totais de processamento até conseguir o resultado final desejado.

No caso, dentre os cenários aplicados, observou-se que o poder de processamento no Google Cloud era ligeiramente superior aos demais e poderia oferecer melhores respostas para nossos cenários com vídeos muito longos, onde o processamento dominava no tempo total. Dessa forma, a alocação de máquinas com capacidade de processamento maiores pode trazer resultados melhores em termos do tempo de processamento. Todavia, alocar máquinas mais potentes em serviços terceiros, agrega um custo maior à manutenção do sistema, dado que os serviços não são gratuitos.

Um outro ponto importante é a proximidade do servidor com o cliente. Quanto mais próximo estiver o servidor em questão, menor será o tempo de comunicação necessário. Analisando os dois servidores utilizados, Google e IC cloud, apesar de em alguns casos o tempo total do Google Cloud ter tido melhor desempenho que o Cloud do IC, se observarmos apenas os tempos de transferências de dados, o Cloud do IC teve tempos de comunicação bem menores. Isso se deve ao fato dos testes terem sido realizados bem mais próximos da infraestrutura do IC Cloud do que do Google Cloud.

O ideal é montar um cenário que se ajuste às necessidades de uma aplicação real que deseja ser implementada. Considerando uma aplicação no modelo cliente-servidor, caso o cliente tenha pouco poder de processamento, ou não possa realizar o processamento dos arquivos por qualquer motivo que seja, os cenários 1 e 3 passam a se tornar inviáveis, deixando mais vantajosos os cenários 2 e 4. E mesmo dentro desses cenários, a análise de proximidade cliente-servidor, bem como custo de alocação das máquinas pode definir qual o cenário mais vantajoso.

Outro fator que pode influenciar na escolha do melhor cenário para implementação é se caso a aplicação apenas precise fazer separações em vídeos pequenos ou vídeos grandes. Para vídeos grandes, a capacidade de processamento tem maior relevância, já para vídeos pequenos o tempo de comunicação ganha maior importância. A implementação de *Fog* pode ser viável em qualquer um dos casos, desde que durante a deliberação certifique-se a presença do Hardware necessário para o suporte, e considerando o custo de alocação das máquinas para este serviço.

Referências

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, 1901.

[2] FERZLI, Rony e KHALIFE, Ibrahim. Mobile Cloud computing educational tool for image/video processing algorithms. Disponível em: <https://cdn.manesht.ir/7715/Mobile%20cloud%20computing%20educational%20tool%20for%20image.pdf>. Acesso em 4 de dez. de 2019.

[3] MESTRE, A. C., CHARÂNTOLA, D., ZANE, R., e BITTENCOURT, L. F. Gerenciamento de Recursos em sistemas distribuídos. Universidade Estadual de Campinas, jul. 2019.

[4] GNU Wget. Free software foundation, Inc., 2018. Disponível em: <https://www.gnu.org/software/wget/>. Acesso em 4 de dez. de 2019.

[5] Moviepy user guide. Zulko, 2017. Disponível em: <http://zulko.github.io/moviepy/>. Acesso em 4 de dez. de 2019.

[6] FICANO, Nick. pytube. Pytube read the docs, 2019. Disponível em: <https://python-pytube.readthedocs.io/en/latest/>. Acesso em 4 de dez. de 2019.

[7] Products and services. Google Cloud, Google Inc., 2019. Disponível em: <https://cloud.google.com/products/>. Acesso em 4 de dez. de 2019.

[8] Compute Engine. Google Cloud, Google Inc., 2019. Disponível em: <https://cloud.google.com/compute/>. Acesso em 4 de dez. de 2019.

[9] Kurose, J. e Ross, K.W., *Computer Networking: A Top-Down Approach*, Fifth Edition, Addison-Wesley, 2009.

[10] Flask. The Pallets Projects. Disponível em: <https://palletsprojects.com/p/flask/>. Acesso em 4 de dez. de 2019.

[11] AWS, Amazon Web Services, Inc., 2019. Disponível em: <https://aws.amazon.com/pt/>. Acesso em 4 de dez. de 2019.

[12] Discover our data center locations. Google Data Centers, Google Inc., 2019. Disponível em: <https://www.google.com/about/datacenters/locations/>. Acesso em 4 de dez. de 2019.

[13] Why Tensorflow. Tensorflow org, 2019. Disponível em: <https://www.tensorflow.org/about/>. Acesso em 4 de dez. de 2019.

[14] UNRUH, Amy. What is the TensorFlow machine intelligence platform. Red Hat, Inc., 2019. Disponível em: <https://opensource.com/article/17/11/intro-tensorflow>. Acesso em 4 de dez. de 2019.

[15] FARHADI, Joseph Redmon Ali. YOLOv3: An Incremental Improvement, University of Washington. 8 abr. 2018.

[16] OLAFENWA, Moses. ImageAI, 2019. Disponível em: <`https://github.com/OlafenwaMoses/ImageAI`>. Acesso em 4 de dez. de 2019.