

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Deaf Accessibility as a Service: uma Arquitetura Escalável e
Tolerante a Falhas para o Sistema de Tradução VLIBRAS

Eduardo de Lucena Falcão

João Pessoa, Paraíba, Brasil
fevereiro de 2014

Universidade Federal da Paraíba
Centro de Informática
Programa de Pós-Graduação em Informática

Deaf Accessibility as a Service: uma Arquitetura Escalável e
Tolerante a Falhas para o Sistema de Tradução VLIBRAS

Eduardo de Lucena Falcão

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática da Universidade Federal da Paraíba como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Computação Distribuída

Alexandre Nóbrega Duarte (Orientador)
Tiago Maritan Ugulino de Araújo (Co-orientador)

João Pessoa, Paraíba, Brasil

©Eduardo de Lucena Falcão, fevereiro de 2014

F178d Falcão, Eduardo de Lucena.
Deaf Accessibility as a Service: uma arquitetura escalável e
tolerante a falhas para o sistema de tradução VLIBRAS /
Eduardo de Lucena Falcão.- João Pessoa, 2014.
141f. : il.
Orientador: Alexandre Nóbrega Duarte
Coorientador: Tiago Maritan Ugulino de Araújo
Dissertação (Mestrado) - UFPB/CI
1. Informática. 2. Computação distribuída. 3. Computação
em nuvem. 4. Processamento paralelo. 5. Sistema de tradução
VLBRAS.

UFPB/BC

CDU: 004(043)

À Deus, e à minha família.

Agradecimentos

Primeiramente à Deus, por me conceder o dom da vida, e me dar força, coragem e perseverança para concluir este trabalho.

Aos meus pais, Marcelo e Virgínia, por sempre me apoiar nessa caminhada, com muito carinho e palavras de incentivo. O que sou hoje é fruto do esforço e dedicação de vocês para me educar. Aos meus irmãos, Fabio e Rafael (e também Thaís e Camila), que sempre me acompanharam de perto e sabem de todas as batalhas que enfrentei nesse mestrado. Obrigado pelos conselhos e pela cumplicidade. Sou reflexo do amor de nossa família.

Ao meu orientador, o professor Alexandre Duarte, por me acolher no mestrado e acreditar no meu potencial. Ao meu co-orientador, professor Tiago Maritan, que muito colaborou e engrandeceu este trabalho. Obrigado pela amizade, pelos ensinamentos, e acima de tudo pelo exemplo de pessoas de caráter no qual espelharei meu futuro profissional.

Um agradecimento especial à Jéssyca e sua família (Angelita, Jennyfer, e Fernando), pelo companheirismo nesses 2 anos de mestrado. Agradeço por sua paciência e amor, nos bons, e principalmente, nos maus momentos. Obrigado por estar sempre ao meu lado.

À um grande companheiro de mestrado, meu amigo Luciano Medeiros, ao qual agradeço de maneira especial por ter contribuído de forma direta neste trabalho com o desenvolvimento do aplicativo VLIBRAS *mobile*. Não apenas por isso, mas pela pessoa simples e humilde que você é, e pelos momentos de descontração que tivemos no mestrado. Muito obrigado!

À Léo Araújo, que muito me ajudou na integração do VLIBRAS com a API DAaaS. Obrigado pelo exemplo de pessoa de garra e perseverança que és!

À João Matos e Carlos Hacks, a quem muito recorri para me auxiliar a compreender assuntos que fugiam da minha área de conhecimento. Também ao Wilter, por me auxiliar em algumas atividades na execução do projeto de experimentos e coleta dos resultados.

Aos amigos do projeto VLIBRAS e do LAVID: Danilo, Vandhuy, Erickson, Lacet, Yúrika, Léo Dantas, Hozana, Virgínia, Manu, Daniel e Fernando. Aos demais amigos de

mestrado: Leandro, Lisieux, Márcio, Ângelo, Berg, André Calisto, Danilo, Wanderson, Amanda, Dália e Andrea. Aos amigos da UFPB Virtual: Marcelle, Anielton, Gláuber, Elba, Edwânia. Obrigado pelo companheirismo nessa caminhada!

Aos meus amigos: Ivan, Bruno Sales, Lucas, Guedes, Erick, Luiz, Gustavo, Rafael, Hermano, Guilherme, Victor, Diego, Ernando, Bruno Rocco, Rafael Rocco, Américo, Múcio, que sempre estarão presentes na minha vida como irmãos. Também aos meus irmãos em Cristo, toda a família Guardiões do Céu, em especial a Mabel e George.

Por fim, meu agradecimento à CAPES pelo auxílio financeiro, e à *Amazon Web Services* pelos créditos para pesquisa que possibilitaram a realização deste trabalho.

Resumo

DEAF ACCESSIBILITY AS A SERVICE

Os surdos enfrentam sérias dificuldades para acessar informações. O fato é que eles se comunicam naturalmente através de línguas de sinais, ao passo que, para a maioria deles, as línguas orais são consideradas apenas uma segunda língua. Quando projetadas, as Tecnologias de Informação e Comunicação (TIC) raramente consideram as barreiras que os surdos enfrentam. É comum que desenvolvedores de aplicações não contratem intérpretes de línguas de sinais para prover uma versão acessível de sua aplicação para surdos. Atualmente existem ferramentas de tradução automática de línguas orais para línguas de sinais, mas, infelizmente, elas não são disponibilizadas à terceiros. Para reduzir esses problemas, seria interessante a disponibilização pública de um serviço de tradução automática entre línguas orais e línguas de sinais. Este é o objetivo geral deste trabalho: utilizar um sistema pré-concebido de tradução automática da língua portuguesa para Língua Brasileira de Sinais (LIBRAS), chamado VLIBRAS, e prover *Deaf Accessibility as a Service*¹ (DAaaS) de forma pública. A ideia é abstrair problemas inerentes no processo de tradução entre a língua portuguesa e LIBRAS através da disponibilização de um serviço que realize a tradução automática de conteúdos multimídia para LIBRAS. O VLIBRAS foi primariamente implantado como um sistema centralizado, e essa arquitetura convencional apresenta algumas desvantagens quando comparada à arquiteturas distribuídas. Neste trabalho, propomos uma arquitetura distribuída para prover este serviço de forma escalável e tolerante a falhas. A escalabilidade e tolerância a falhas da solução proposta foi validada através de um projeto de experimentos. Para concepção deste serviço, é utilizado o paradigma de computação em nuvem para incorporar os seguintes benefícios adicionais: transparência, alta disponibilidade, e uso eficiente dos recursos.

Palavras-chave: computação em nuvem, processamento paralelo, acessibilidade.

¹ Acessibilidade para Surdos como um Serviço

Abstract

DEAF ACCESSIBILITY AS A SERVICE

Deaf people face serious difficulties to access information. The fact is that they communicate naturally through sign languages, whereas, to most of them, the spoken languages are considered only a second language. When designed, Information and Communication Technologies rarely take into account the barriers that deaf people face. It is common that application developers do not hire sign languages interpreters to provide an accessible version of their application to deaf people. Currently, there are tools for automatic translation from spoken languages to sign languages, but, unfortunately, they are not available to third parties. To reduce these problems, it would be interesting if any automatic translation service could be publicly available. This is the general goal of this work: use a preconceived machine translation from portuguese language to Brazilian Sign Language (LIBRAS), named VLIBRAS, and provide Deaf Accessibility as a Service (DAaaS) publicly. The idea is to abstract inherent problems in the translation process between the portuguese language and LIBRAS by providing a service that performs the automatic translation of multimedia content to LIBRAS. VLIBRAS was primarily deployed as a centralized system, and this conventional architecture has some disadvantages when compared to distributed architectures. In this paper we propose two distributed architectures in order to provide a scalable service and achieve fault tolerance. Scalability and fault tolerance were validated through experiments. For conception of this service, it is used the cloud computing paradigm to incorporate the following additional benefits: transparency, high availability, and efficient use of resources.

Keywords: cloud computing, parallel processing, accessibility.

Conteúdo

1	Introdução	1
1.1	Motivação	2
1.2	Objetivos	5
1.3	Metodologia	6
1.4	Estrutura da Dissertação	7
2	Fundamentação Teórica	8
2.1	Línguas de Sinais	8
2.2	LIBRAS	9
2.3	Sistema de Tradução Automática	10
2.3.1	Sistema de Tradução Automática VLIBRAS	11
2.4	Sistemas Distribuídos	13
2.5	Computação em Nuvem	18
2.5.1	Características Essenciais	20
2.5.2	Modelos de Serviço	22
2.5.3	Modelos de Implantação	25
2.5.4	Balanceamento de Carga e Provisionamento Dinâmico de Recursos	27
2.5.5	<i>Amazon Web Services</i>	27
2.6	Considerações	32
3	Trabalhos Relacionados	33
3.1	Acessibilidade para Surdos Brasileiros nas TICs	33
3.1.1	Rybená	34
3.1.2	F-LIBRAS	35

3.1.3	FALIBRAS	35
3.1.4	TLIBRAS	35
3.1.5	VE-LIBRAS	36
3.1.6	POLI-LIBRAS	36
3.1.7	ProDeaf	36
3.1.8	VLIBRAS	37
3.1.9	Síntese	37
3.2	Computação em Nuvem e Processamento Multimídia	39
3.2.1	Escalabilidade na Nuvem	39
3.2.2	Tolerância a Falhas na Nuvem	41
3.2.3	Sistemas de Processamento Multimídia Implantados na Nuvem	43
3.3	Considerações	47
4	Arquitetura Proposta	48
4.1	VLIBRAS - Arquitetura Centralizada	49
4.2	DAaaS - Arquitetura Distribuída	50
4.2.1	Tolerância a Falhas	54
4.2.2	Provisionamento Dinâmico de Recursos	56
4.3	API DAaaS	59
4.4	Cenários de Uso da API	61
4.5	Considerações	62
5	Experimentos	65
5.1	Carga de Trabalho	65
5.2	Testes Preliminares	66
5.3	Projeto de Experimentos	74
5.3.1	Execução do Projeto de Experimentos	75
5.4	Resultados	77
5.5	Considerações	83
6	Considerações Finais	85
	Referências Bibliográficas	91

<i>CONTEÚDO</i>	<i>x</i>
A API DAaaS - Especificação JSON	92
B Carga de Trabalho dos experimentos	95
C Tabelas com Resultados dos Experimentos Preliminares	97
D Projeto de Experimentos	102
E Configuração do Ambiente de Experimentos	120

Lista de Símbolos

ABNT : Associação Brasileira de Normas Técnicas

AMI : *Amazon Machine Image*

API : *Application Programming Interface*

ASL : *American Sign Language*

AWS : *Amazon Web Services*

BPMN : *Business Process Modeling Notation*

BSL : *British Sign Language*

CAPES : Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

DAaaS : *Deaf Accessibility as a Service*

DGE : Departamento de Governo Eletrônico

e-MAG : Modelo de Acessibilidade em Governo Eletrônico

Ec2 : *Elastic Compute Cloud*

ECU : *Elastic Computing Unit*

ELB : *Elastic Load Balancing*

FSL : *French Sign Language*

GB : *GigaByte*

GHz : *GigaHertz*

HD : *Hard Disk*

HTTP : *Hypertext Transfer Protocol*

IaaS : *Infrastructure as a Service*

IBGE : Instituto Brasileiro de Geografia e Estatística

ISO : *International Organization for Standardization*

IP : *Internet Protocol*

JSON : *JavaScript Object Notation*

LAVID : Laboratório de Aplicações de Vídeo Digital

LIBRAS : Língua Brasileira de Sinais

LS : Língua de Sinais

LSKB : Língua de Sinais Kaapor Brasileira

MinC : Ministério da Cultura

MB : *MegaByte*

NIST : *National Institute of Standards and Technology*

O : Objeto

PaaS : *Platform as a Service*

PC : *Personal Computer*

PPGI : Programa de Pós-Graduação em Informática

QoS : *Quality of Service*

RAM : *Random Access Memory*

REST : *Representational state transfer*

RNP : Rede Nacional de Ensino e Pesquisa

S : Sujeito

S3 : *Simple Storage Service*

SaaS : *Software as a Service*

SSD : *Solid State Disk*

SENAPES : Seminário Nacional de Acessibilidade para Pessoas Surdas

SINTRA : Sindicato Nacional dos Tradutores

SLA : *Service Level Agreement*

SMS : *Short Message Service*

SNS : *Simple Notification Service*

SQS : *Simple Queue Service*

SSD : *Solid State Disks*

SSL : *Spanish Sign Language*

SVO : sujeito-verbo-objeto

TI : Tecnologia da Informação

TIC : Tecnologia de Informação e Comunicação

UFPB : Universidade Federal da Paraíba

V : Verbo

VoD : *Video on Demand*

VRML :

X3D :

XML : *eXtensible Markup Language*

WHO : *World Health Organization*

Lista de Figuras

2.1	Arquitetura do sistema VLIBRAS	12
2.2	Tipos de escalonamento	16
2.3	Virtualização	19
2.4	Empresa de TI que não utiliza o paradigma de computação em nuvem (SOUSA, 2013)	20
2.5	Disponibilização de recursos de maneira elástica e escalável (SOUSA, 2013)	21
2.6	Modelos de serviço (SOUSA, 2013)	24
2.7	Modelos de implantação	27
2.8	Balanceamento de carga aliado ao provisionamento dinâmico de recursos .	28
3.1	Paralelismo a nível de usuário (ZHU et al., 2011)	44
3.2	Paralelismo a nível de tarefa (ZHU et al., 2011)	45
4.1	Arquitetura do VLIBRAS centralizada em um único servidor	51
4.2	DAaaS - Arquitetura Distribuída	53
4.3	Modelo de Processos de Negócios para Tolerância a Falhas Reativa	57
4.4	Modelo de Processos de Negócios para Provisionamento Dinâmico de Re- cursos	59
4.5	Selecionando o texto	61
4.6	Clicando no botão “Traduzir”	61
4.7	Avatar traduzindo o texto selecionado	62
4.8	Tela principal	63
4.9	Opções	63
4.10	Opção: texto	63
4.11	Processando	63

4.12	Concluída	63
4.13	Tocar tradução	63
4.14	Avatar traduzindo o texto selecionado	64
5.1	Influência do Fator B na Variação dos Tempos de Tradução	81
5.2	Influência do Fator D na Variação dos Tempos de Tradução	82
5.3	Custo das instâncias Ec2 nos 16 experimentos	83
5.4	Influência do Fator C na Variação dos Tempos de Tradução	83
5.5	Influência do Fator A na Variação dos Tempos de Tradução	84

Lista de Tabelas

1.1	Média de tempo de processamento para requisições concorrentes	5
3.1	Sistemas de Tradução Automática Português -> LIBRAS. Adaptado de: (PI-VETTA; ULBRICHT; SAVI, 2011).	38
4.1	Tipos de Traduções disponíveis no VLIBRAS, e seus <i>status</i> na API DAaaS	60
5.1	Configuração de <i>hardware</i> das instâncias	67
5.2	Tempos de processamento dos componentes em três instâncias diferentes .	69
5.3	Instância <i>m1.small</i> HD	71
5.4	Texto	71
5.5	Legenda	71
5.6	Instância <i>c3.xlarge</i> SSD	72
5.7	Texto	72
5.8	Legenda	72
5.9	Valores escolhidos nos experimentos preliminares	73
5.10	Fatores do projeto fatorial $2^k r$	75
5.11	Projeto de Experimentos - Resultados	78
5.12	Projeto de Experimentos - Influência dos Fatores na Variação dos Resultados Coletados	80
C.1	Média de tempo de processamento para traduções concorrentes de texto - <i>m1.small</i>	98
C.2	Média de tempo de processamento para traduções concorrentes de legenda - <i>m1.small</i>	99

C.3	Média de tempo de processamento para traduções concorrentes de texto - c3.xlarge	100
C.4	Média de tempo de processamento para traduções concorrentes de legenda - c3.xlarge	101
D.1	Projeto de Experimentos	103
D.2	$A = -1, B = -1, C = -1, D = -1$	104
D.3	$A = 1, B = -1, C = -1, D = -1$	105
D.4	$A = -1, B = 1, C = -1, D = -1$	106
D.5	$A = -1, B = -1, C = 1, D = -1$	107
D.6	$A = -1, B = -1, C = -1, D = 1$	108
D.7	$A = 1, B = 1, C = -1, D = -1$	109
D.8	$A = -1, B = 1, C = 1, D = -1$	110
D.9	$A = -1, B = -1, C = 1, D = 1$	111
D.10	$A = 1, B = -1, C = 1, D = -1$	112
D.11	$A = 1, B = -1, C = -1, D = 1$	113
D.12	$A = -1, B = 1, C = -1, D = 1$	114
D.13	$A = 1, B = 1, C = 1, D = -1$	115
D.14	$A = -1, B = 1, C = 1, D = 1$	116
D.15	$A = 1, B = -1, C = 1, D = 1$	117
D.16	$A = 1, B = 1, C = -1, D = 1$	118
D.17	$A = 1, B = 1, C = 1, D = 1$	119
E.1	Parâmetros utilizados no ab	121

Lista de Códigos Fonte

4.1	Exemplo de uma requisição HTTP POST ao DAaaS	60
5.1	Injeção de 10% de falhas no VLIBRAS	76
A.1	Exemplo de JSON para requisições de tradução de texto	92
A.2	Exemplo de JSON para requisições de tradução de legenda	92
A.3	Exemplo de JSON para requisições de tradução do áudio	92
A.4	Exemplo de JSON para requisições de tradução do áudio do vídeo	92
A.5	Exemplo de JSON para requisições de tradução do <i>closed caption</i>	93
A.6	Exemplo de JSON para requisições de tradução de legenda com mixagem do resultado com um vídeo	93
A.7	Exemplo de JSON para requisições de tradução do áudio do vídeo com mi- xagem do resultado com o vídeo	93
A.8	Exemplo de JSON para requisições de tradução do <i>closed caption</i> do vídeo com mixagem do resultado com o vídeo	94
E.1	Arquivo de configuração do Tomcat 7 (/etc/default/tomcat7)	120
E.2	Linha de comando do ab para simular o 1º experimento para A(−1)/B(1)/C(−1)/D(−1)	120
E.3	Arquivo de saída do ab	120

Capítulo 1

Introdução

É intrínseco ao ser humano a necessidade de se comunicar e expressar. Segundo Russell e Norvig (2004), “a comunicação é a troca intencional de informações provocada pela produção e percepção de sinais extraídos de um sistema compartilhado de sinais convencionais”. Através desses sistemas compartilhados de sinais, denominados línguas, é que torna-se possível a transmissão de informações entre indivíduos.

A língua que o indivíduo usa para se comunicar depende de sua natureza além do grupo de indivíduos com o qual ele convive. Os ouvintes, por exemplo, comunicam-se por intermédio de línguas oralizadas, ou seja, através de sons articulados que são percebidos pelo sistema auditivo. Já os surdos, por outro lado, encontraram na linguagem gestual-corporal um meio eficaz de comunicação como alternativa à falta de capacidade auditiva. Essa modalidade, denominada língua de sinais (LS), envolve elementos lingüísticos manuais, corporais e faciais para articular os sinais que são compreendidos através do sistema visual. Portanto, a língua na qual o surdo consegue perceber e produzir de maneira natural é a LS, ao passo que as línguas orais, utilizadas cotidianamente pela maioria das pessoas e em praticamente todos os meios de comunicação, representam apenas “uma segunda língua” (GÓES, 1996).

Normalmente, nos diferentes contextos da sociedade atual brasileira, a informação é transmitida através da língua portuguesa. Quando os contextos são as TICs em conjunto com a Internet, é preciso ressaltar que se o usuário é surdo, ele é um indivíduo bilíngue cuja língua primária é a LIBRAS, e a língua portuguesa é apenas sua segunda língua. Portanto, o nível de proficiência deste indivíduo na língua portuguesa pode tornar a leitura uma tarefa árdua e limitada (GOMES; GÓES, 2011), fazendo deste fator uma barreira a mais na inclusão

digital.

Segundo o censo demográfico realizado em 2010 pelo Instituto Brasileiro de Geografia e Estatística (IBGE), cerca de 9,7 milhões de brasileiros possuem deficiência auditiva, o que representa 5,1% da população. Deste total, cerca de 2 milhões possuem a deficiência auditiva severa - 1,7 milhões têm grande dificuldade para ouvir e 344,2 mil são surdos - e 7,5 milhões apresentam alguma dificuldade auditiva (IBGE, 2010). Em termos mundiais, a estimativa da Organização Mundial de Saúde é de que aproximadamente 360 milhões de pessoas apresentem algum nível de deficiência auditiva (WHO, 2013). Isso implica que os surdos representam uma parcela significativa da população brasileira e mundial.

Com objetivo de minimizar estes problemas, o presente trabalho visa prover às diferentes TICs um serviço automatizado de tradução de conteúdos multimídia da língua portuguesa para LIBRAS. Para tanto, um sistema pré-concebido de tradução automática da língua portuguesa para LIBRAS, chamado VLIBRAS, será utilizado como base. Atualmente, o VLIBRAS não é disponibilizado publicamente nem consegue atender grandes cargas de requisições. Portanto, a principal proposta deste trabalho é projetar e validar uma arquitetura para prover *Deaf Accessibility as a Service* (DAaaS). A ideia é utilizar o paradigma de computação em nuvem para prover o DAaaS à terceiros de forma transparente, escalável, e tolerante a falhas.

1.1 Motivação

Com a rápida evolução das TICs nas duas últimas décadas, o governo brasileiro vem buscando meios de implementar políticas de inclusão social com relação à problemática de acessibilidade na Internet. A Lei 10.098, instituída no ano 2000, estabelece normas gerais e critérios básicos para a promoção da acessibilidade das pessoas com deficiência (Lei número 10.098, 2000). Apesar de mostrar-se preocupada com a dificuldade de acesso a informação por parte destas pessoas, a lei não preenche determinadas lacunas de maneira incisiva, como por exemplo, a acessibilidade à Internet.

O próximo passo do governo brasileiro foi a elaboração do modelo de acessibilidade do governo eletrônico, para facilitar o acesso de todas as pessoas às informações e serviços disponibilizados nos portais do governo. Deste modo, a primeira versão do e-MAG (Modelo de

Acessibilidade em Governo Eletrônico) foi disponibilizada para consulta pública em 18 de janeiro de 2005. Em 2007, a Portaria nº 3, de 7 de maio, institucionalizou o e-MAG, tornando sua observância obrigatória nos portais do governo brasileiro (DGE, 2011). Paralelamente, a ideia era reutilizar este modelo de acessibilidade e-MAG por meio da disponibilização de uma cartilha técnica para os desenvolvedores de sítios convencionais, apresentando detalhadamente a proposta de implementação das recomendações de acessibilidade em portais do governo.

A redação do documento de referência para o e-MAG versão 2.0, lançado em dezembro de 2005, apresentou algumas limitações quando tratou de acessibilidade para usuários surdos. Com relação a estes usuários, o DGE (2005) delimita as seguintes recomendações para o nível de prioridade I (mais importante):

1. “Utilizar a linguagem mais clara e simples possível, logicamente, adequada ao conteúdo do sítio”;
2. “A faixa que contém a legenda é uma alternativa para espectadores surdos ou com dificuldades auditivas”;
3. “... a transcrição do áudio e do áudio-descrição continuam a ser as melhores alternativas”.

Sabe-se que os surdos se comunicam naturalmente através de línguas de sinais, sendo as orais apenas uma segunda língua. É importante considerar que a maioria dos surdos passa vários anos na escola mas não consegue atingir proficiência na leitura e escrita da língua oral de seu país, justamente pelo fato de tais línguas possuírem grafias baseadas em sons (STUMPF, 2000). O documento do e-MAG versão 2.0 (DGE, 2005) cita unicamente a transcrição do áudio em faixas de legenda, ou simplesmente textos, como alternativa para a inclusão de surdos nos ambientes digitais. Diante do exposto, é interessante ressaltar os resultados de um estudo realizado por Wauters (2005), tendo como público-alvo crianças e adolescentes surdos holandeses de 7 a 20 anos de idade, que relata que apenas 25% deles possuem capacidade de leitura igual ou superior ao de uma criança sem deficiência de 9 anos. Portanto, se o texto for extremamente simples e claro, essa seria uma alternativa razoável para os surdos, mas não a única e tampouco a ideal.

Desde o lançamento do e-MAG versão 2.0, estudiosos da área vem discutindo e reforçando que a acessibilidade para surdos não deve considerar apenas a garantia de legendas ou descrições para acesso a conteúdo sonoro, mas prioritariamente a tradução em LIBRAS de páginas e conteúdos da Web (GOMES; GÓES, 2011). Desse modo, ao reformular e aprimorar o e-MAG para a versão 3.0, o DGE (2011) afirma que “além de alternativa em texto e legenda, é desejável que os vídeos com áudio apresentem alternativa na língua brasileira de sinais”. É importante a tradução em LIBRAS não somente para os áudios de vídeos, mas também para os textos mais complexos dos sítios, uma vez que a língua portuguesa não é a primeira língua dos surdos.

A tradução para LIBRAS é de extrema importância, mas traz consigo alguns problemas:

1. Alto custo do serviço: segundo o SINTRA (2013), a tradução de 60 minutos de um áudio em língua portuguesa para a LIBRAS custa 585 reais;
2. Grande dinamismo de conteúdos da Internet: a frequência com que novas páginas são inseridas na Web é intensa, assim como é a frequência de atualização dos conteúdos das mesmas.

Para os sistemas de vídeo sob demanda (*Video on Demand* - VoD), prover acessibilidade para pessoas surdas também é uma tarefa complicada. O *YouTube*, por exemplo, buscou tornar seus vídeos acessíveis para surdos através da disponibilização de legendas. Deve ser ressaltada a inviabilidade da tradução de seus vídeos por intérpretes, devido aos milhões de usuários do serviço, e à grande quantidade de vídeos que são enviadas ao sistema. Os gastos seriam extremamente elevados, e não haveriam intérpretes suficientes para a demanda.

Uma das melhores soluções para sistemas de VoD seria a utilização de um serviço de tradução automática da língua portuguesa para a LIBRAS. Para tanto, o sistema VLIBRAS pode ser utilizado como base. A principal motivação para a escolha do sistema VLIBRAS é sua arquitetura flexível, que permite sua implantação em diferentes plataformas (PC, Web, TV Digital), e seu potencial de gerar a tradução para diferentes tipos de conteúdos, e.g., texto, áudio e vídeo (legenda, *closed caption*, e áudio).

Atualmente, o VLIBRAS é dotado de uma infraestrutura convencional, sendo implantado em uma arquitetura centralizada. Contudo, sabe-se que sistemas centralizados apresentam várias desvantagens quando comparados à sistemas distribuídos. A principal desvantagem é

a incapacidade de alocar novos recursos quando a demanda aumenta, convergindo para uma má oferta do serviço e maior possibilidade de falhas no processamento das requisições. Para termos uma noção inicial da performance do VLIBRAS implantado em uma arquitetura centralizada¹, medimos o tempo de processamento para 1, 10, 20, e 40 requisições simultâneas (1.1). Tais requisições foram compostas por textos com 2202 caracteres que constituem 324 palavras.

Tabela 1.1: Média de tempo de processamento para requisições concorrentes

Nº de requisições	Tempo médio	Pior tempo	Desvio padrão	Falhas
1	21.369s	21.369s	0	0
10	109.618s	118.023s	21.682	0
20	206.45106	223.023s	49.490	0
40	279.612s	281.016s	2.153	7

Com esses testes, foi possível constatar que o sistema falha ao atender um número de requisições igual ou superior a 40. Deste modo, fica mais claro entender o motivo da necessidade de uma arquitetura distribuída e dinâmica. Como a API será disponibilizada de forma pública e irrestrita², em momentos de pico o sistema poderá receber cargas muito superiores a 40 requisições, porém, em momentos de baixa demanda, também poderá receber um número de requisições inferior a 40. Uma das principais características da arquitetura distribuída que será proposta é a capacidade de provisionamento dinâmico de recursos. Com ela, é possível economizar financeiramente em momentos de baixa demanda, mas também prover um serviço de qualidade mesmo quando submetidos a grandes cargas de requisições.

1.2 Objetivos

Uma vez demonstrado nas seções anteriores o contexto geral da problemática que os surdos enfrentam enquanto navegam na Internet, este trabalho tem como objetivo geral a oferta de DAaaS - um serviço de acessibilidade para surdos. Para tanto, será disponibilizado a quem interessar, um serviço automatizado de tradução de conteúdos multimídia para a LIBRAS.

¹Instância da Amazon c3.large, Ubuntu 12.04 - 64 bits, 2 vCPU (CPUs virtuais) e 7 ECUs (unidade de processamento elástico), 3.75GB RAM

²Enquanto a AWS disponibilizar créditos.

A ideia é utilizar o paradigma de computação em nuvem para permitir que os usuários (e.g., sistemas de VoD, desenvolvedores de sítios e aplicativos) possam tornar seus conteúdos acessíveis de forma transparente, sem preocupar-se em como ocorre o processo de tradução e atividades relacionadas. Além da transparência na oferta do serviço, o paradigma de computação em nuvem possibilita que o mesmo seja escalável, tolerante a falhas, e faça do uso eficiente de recursos.

Como forma de validar o objetivo principal deste trabalho, os seguintes objetivos específicos foram definidos:

Objetivo 1: propor uma arquitetura escalável e tolerante a falhas para um sistema de processamento multimídia;

Objetivo 2: validar a arquitetura através de um projeto de experimentos, e por intermédio da disponibilização da API para algum sítio da Internet ou sistema de VoD, visando permitir que qualquer usuário surdo tenha acesso ao conteúdo do mesmo em LIBRAS.

O presente trabalho utilizará como base o sistema de tradução VLIBRAS. É importante ressaltar que a avaliação da qualidade de tradução do VLIBRAS não está no escopo deste trabalho, devido a sua complexidade.

1.3 Metodologia

A elaboração deste trabalho tem como metodologia as seguintes atividades:

Atividade 1 - Análise bibliográfica: realizar um levantamento bibliográfico detalhado sobre os principais trabalhos relacionados à disponibilização de acessibilidade para surdos brasileiros nas TICs. Pesquisar os trabalhos mais relevantes que envolvam computação em nuvem, focando em escalabilidade e tolerância a falhas, de preferência para processamento multimídia;

Atividade 2 - Estudo sobre o paradigma de computação em nuvem: realizar um estudo sobre técnicas de desenvolvimento e disponibilização de serviços utilizando a infraestrutura de computação em nuvem, com foco prático na *Amazon Web Services*

(AWS), uma vez que esta empresa disponibilizou créditos para o desenvolvimento deste trabalho;

Atividade 3 - Planejamento, implementação e implantação da arquitetura na nuvem: planejar a adaptação da arquitetura do VLIBRAS (antes centralizada) para uma arquitetura distribuída a ser implantada na nuvem;

Atividade 4 - Validação do serviço:

Experimentos: realizar um conjunto de experimentos para validar o serviço quanto a escalabilidade e capacidade de tolerância a falhas;

Utilização do serviço: utilizar o serviço/API de forma experimental em um sítio da Internet e aplicativo *mobile*.

1.4 Estrutura da Dissertação

Este documento está dividido em 6 Capítulos. O Capítulo 1 apresenta o trabalho de maneira geral descrevendo sua motivação, relevância, contribuição, objetivos, metodologia e estrutura da dissertação. No Capítulo 2, se encontra a fundamentação teórica com as definições relacionadas a línguas de sinais, LIBRAS, sistema de tradução automática VLIBRAS, sistemas distribuídos e computação em nuvem, destacando os principais conceitos e técnicas necessárias ao desenvolvimento do DAaaS. O Capítulo 3 apresenta os principais trabalhos relacionados à disponibilização de acessibilidade para surdos brasileiros nas TICs, além dos trabalhos que abordem técnicas de computação em nuvem para oferecer escalabilidade e tolerância a falhas, e sistemas de processamento multimídia implantados na nuvem. Detalhes da arquitetura proposta bem como de sua implementação são descritos no Capítulo 4. No Capítulo 5, encontra-se o projeto de experimentos, com seus respectivos resultados, para avaliação e validação da capacidade de escalonamento e tolerância a falhas. O Capítulo 6 fecha o documento com as conclusões, discussão, trabalhos futuros e considerações finais. As informações complementares do trabalho são apresentadas nos Apêndices e nos Anexos deste documento.

Capítulo 2

Fundamentação Teórica

Nesse Capítulo serão apresentados os principais conceitos que fundamentam este trabalho. Inicialmente, serão descritas algumas características gerais das línguas de sinais, e em seguida serão expostas as principais características específicas, propriedades e conceitos relacionados à língua brasileira de sinais. Posteriormente, será detalhada a arquitetura e funcionamento do sistema de tradução automática VLIBRAS. Por fim, os principais conceitos relacionados a sistemas distribuídos e computação em nuvem serão apresentados.

2.1 Línguas de Sinais

A língua de sinais é considerada a língua natural dos surdos (ALMEIDA, 2000). Brito (1995) explica que elas são consideradas línguas naturais, pois surgem espontaneamente da interação entre os deficientes auditivos, provendo aos mesmos a capacidade de expressar qualquer conceito descritivo, concreto, racional, literal, metafórico, emocional ou abstrato.

A língua de sinais é dotada de características peculiares: utiliza-se de gestos e expressões faciais como canal de comunicação substituto à vocalização (MEIRELLES; GALVÃO, 2004). Tais línguas possuem gramática própria e são compostas por níveis linguísticos como fonologia, morfologia, sintaxe e semântica (BRITO, 1995). Similarmente às línguas orais, línguas de sinais também possuem itens léxicos: os sinais (STOKOE, 1980).

2.2 LIBRAS

No mundo existem diversas línguas de sinais, cada uma contendo suas próprias regras gramaticais, vocabulários e fonemas (BUTTUSSI; CHITTARO; COPPO, 2007). Como exemplo, podem ser citadas a língua americana de sinais (*American Sign Language* - ASL) para os Estados Unidos (STOKOE, 1980) (MEIRELLES; GALVÃO, 2004), a língua britânica de sinais (*British Sign Language* - BSL) para a Inglaterra (STOKOE, 1980), a língua de sinais espanhola (*Spanish Sign Language* - SSL) para a Espanha (SAN-SEGUNDO et al., 2006), e a língua francesa de sinais (*French Sign Language* - FSL) para a França (STOKOE, 1980). Entretanto, é normal que alguns países tenham mais de uma língua de sinais. No Brasil, por exemplo, a maioria dos deficientes auditivos utiliza a língua brasileira de sinais (que é reconhecida pela lei brasileira 10.436), ao passo em que uma pequena comunidade indígena da floresta amazônica brasileira utiliza a língua de sinais kaapor brasileira (LSKB) (SILVA, 2012). Dentro do contexto da LIBRAS, é interessante considerar que apesar de ser a LS oficial, existem pequenas variações entre os estados brasileiros, os denominados regionalismos.

Na LIBRAS, os sinais são formados pelos mesmos fonemas das demais LS. Entretanto, a LIBRAS é dotada de uma gramática própria, diferente da gramática da língua portuguesa. Com relação à ordem das palavras, por exemplo, existem diferenças entre a língua portuguesa e a LIBRAS (ARAÚJO, 2012). Na maioria dos casos, a língua portuguesa utiliza sentenças no formato sujeito-verbo-objeto (SVO), enquanto que a LIBRAS geralmente utiliza sentenças no formato tópico-comentário (BRITO, 1995). Araújo (2012) utilizou os seguintes exemplos para explicar esta diferença:

- O urso (S) matou (V) o leão (O).
- Eu (S) não vi (V) o acidente na rua (O).

As sentenças supracitadas seriam representadas em LIBRAS da seguinte forma:

- URSO (Tópico), LEÃO MATAR (Comentário).
- RUA ACIDENTE (Tópico) NÃO ENXERGAR (Comentário).

Apesar da ordem dos argumentos na estrutura das sentenças em LIBRAS ser diferente da ordem das mesmas na língua portuguesa, existem algumas semelhanças na estrutura das

sentenças. Segundo Brito (1995), em ambas as línguas, “toda sentença possui um núcleo que é o elemento que possui valência”. Tanto na língua portuguesa quanto na LIBRAS, o verbo é o elemento que possui valência e dita o número e o tipo de argumentos ou complementos necessários. Ao analisar o verbo “enviar”, pode ser percebido que tanto na língua portuguesa como na LIBRAS ambos possuem a mesma valência, pois necessitam de três argumentos (ARAÚJO, 2012). Araújo (2012) exemplifica tal fato com as seguintes sentenças:

- Paulo enviou o livro ao amigo. (em língua portuguesa)
- LIVRO AMIGO P-A-U-L-O ENVIAR. (em LIBRAS)

É possível observar, nos dois exemplos, que independentemente da ordem das palavras, as sentenças são constituídas de um núcleo (o verbo enviar) e três argumentos ou complementos (Paulo, amigo e livro). Outra característica relevante é que em LIBRAS, os nomes próprios são representados soletrando-se as letras (e.g.: o nome Paulo é representado em LIBRAS como P-A-U-L-O) (ARAÚJO, 2012).

Na próxima seção será apresentado o funcionamento do sistema de tradução automática utilizado no presente trabalho.

2.3 Sistema de Tradução Automática

A tradução automática é o ato de converter conteúdos entre línguas naturais através de sistemas computacionais. Contudo, o processo de tradução é dotado de algumas dificuldades e desafios inerentes ao problema. Um problema comum na tradução automática, é o contexto em que aquele conteúdo transmitido está inserido. Quando uma mensagem é transmitida entre dois ou mais interlocutores, é preciso que se perceba o conjunto de conhecimentos de senso comum implícitos para tratá-los pelo sistema computacional e conseguir gerar uma boa tradução (ARAÚJO, 2012). Adicionalmente, existe um conjunto de ambiguidades (ambiguidade léxica, sintática, semântica, contextual) intrínseco às linguagens naturais que também precisam ser tratadas pelo sistema de tradução automática (DORR; JORDAN; BENOIT, 1999).

É importante ressaltar que o objetivo do presente trabalho é prover por meio de um sistema distribuído (utilizando o paradigma de computação em nuvem) um serviço de tradução

automatizada para vídeos da língua portuguesa para a LIBRAS. Apesar deste trabalho utilizar um sistema de tradução previamente concebido, avaliar o nível de acerto na tradução realizada pelo sistema não entra no escopo dos objetivos delimitados.

2.3.1 Sistema de Tradução Automática VLIBRAS

O sistema denominado por VLIBRAS é composto por um conjunto de componentes responsáveis por gerar automaticamente (ou seja, sem intervenção humana direta) a tradução em LIBRAS a partir dos recursos disponíveis nesses conteúdos (texto, áudio, vídeo, ou legenda). O funcionamento e arquitetura do sistema será detalhado de maneira sequencial, de acordo com a a Figura 2.1.

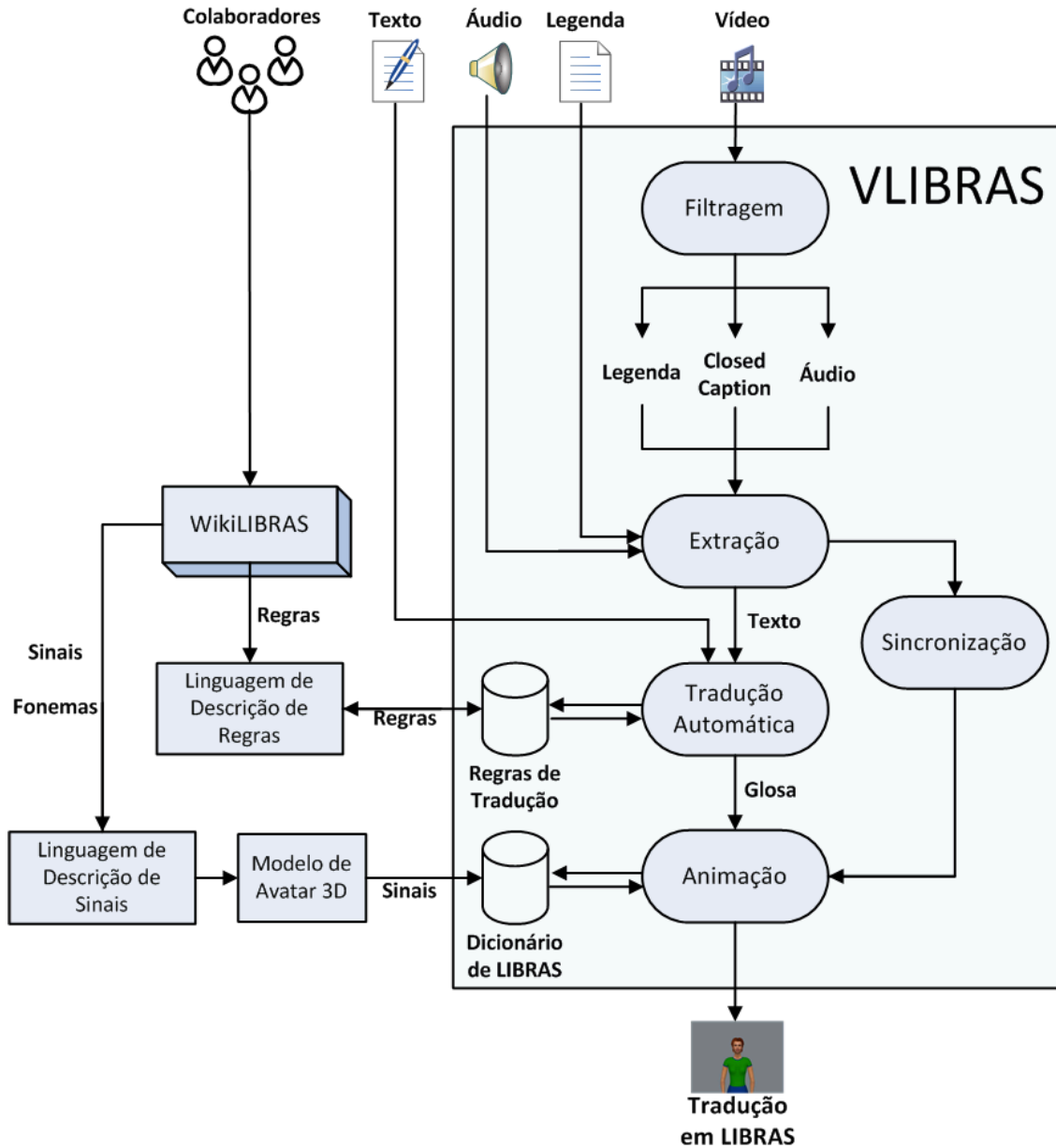
O VLIBRAS pode receber quatro formas diferentes de entrada: um arquivo de vídeo, um arquivo de legenda, um arquivo em formato de áudio, ou um arquivo com texto puro. Se o sistema recebe um vídeo, o mesmo é submetido a um componente de Filtragem, responsável por extrair as faixas de legendas, *closed caption*, ou áudio, embutidas nesses conteúdos. Opcionalmente, um arquivo de legenda ou áudio pode ser carregado diretamente na solução, o que dispensa a etapa de Filtragem. É possível, também, submeter um arquivo contendo apenas texto, o que elimina as etapas de Filtragem e de Extração. O componente de Extração converte as faixas de legendas, *closed caption*, ou áudio em uma sequência de palavras em língua portuguesa. O componente de Extração ainda é responsável por obter e transmitir informações de tempo ao componente de Sincronização.

O componente de Tradução Automática recebe um conjunto de textos como entrada do componente de Extração. Essa sequência de palavras é, portanto, automaticamente traduzida para uma sequência de glosas em LIBRAS. A estratégia de tradução de texto para glosa¹ foi projetada para traduzir conteúdos de forma eficiente e para domínios gerais, além de combinar métodos de compressão estatística utilizados para classificar os tokens (palavras) de entrada, estratégias de simplificação textual para reduzir a complexidade do texto de entrada, e um conjunto de regras morfológicas e sintáticas definido por especialistas.

A sequência de glosas gerada pelo componente de Tradução Automática é, então, enviada para um componente de Animação que associa cada glosa com uma representação visual de um sinal (vídeo do avatar 3D) no Dicionário de LIBRAS. O componente de Animação

¹Glosa: representação textual de LIBRAS.

Figura 2.1: Arquitetura do sistema VLIBRAS



também recebe como entrada os pontos de sincronização informados pelo componente de Sincronização, para que a tradução tenha sintonia com o conteúdo multimídia original. Por fim, a sequência de glosas é mapeada para uma sequência de vídeos dos sinais, levando em consideração as etiquetas de tempo previamente definidas e resultando, portanto, em um vídeo de tradução em LIBRAS para o conteúdo de entrada.

É interessante ressaltar que os dicionários de LIBRAS são utilizados para evitar a renderização dos sinais em tempo real, uma vez que essa tarefa consome muito tempo. Tais

dicionários armazenam vídeos dos sinais de LIBRAS pré-renderizados e cada sinal possui um código (por exemplo, sua representação textual em glosa) associado com esse vídeo. Dessa forma, é possível gerar um vídeo de LIBRAS a partir da combinação de sinais no dicionário de LIBRAS. Outro aspecto importante da solução é a utilização de estratégias de colaboração e computação humana para desenvolver as construções linguísticas da solução de forma eficiente e semi-automática. A ideia dessa abordagem é que especialistas em LIBRAS colaborem na geração dessas construções linguísticas e também melhorem a qualidade dos conteúdos gerados através da melhoria das regras de tradução, da inclusão de novos sinais, etc. Para isso, uma ferramenta de computação humana, denominada WikiLIBRAS, foi desenvolvida, juntamente com linguagens formais para descrever regras de tradução (Linguagem de Descrição de Regras de Tradução) e sinais (Linguagens de Descrição de Sinais), e o modelo de um agente animado virtual 3D (avatar 3D).

O sistema VLIBRAS vem sendo aprimorado há alguns anos pela equipe do LAVID, através de apoios financeiros da RNP, CAPES e MinC. Araújo (2012), Ferreira (2012) e Silva (2012) são alguns dos trabalhos que contribuíram para o aperfeiçoamento do sistema.

Na próxima seção serão apresentados os principais conceitos relacionados a sistemas distribuídos e computação em nuvem que são importantes no contexto do presente trabalho.

2.4 Sistemas Distribuídos

Nos últimos anos, a demanda por recursos computacionais tem ultrapassado os limites de poder de processamento que um computador pode oferecer. Nesse sentido, tem se tornado frequente a implementação de sistemas distribuídos com objetivo de compensar tais disparidades. As duas definições mais comumente utilizadas para sistemas distribuídos são de Tanenbaum e Steen (2007) e Coulouris (2009). Tanenbaum e Steen (2007) afirmou que “um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente”, e Coulouris (2009) definiu sistemas distribuídos como uma “coleção de computadores autônomos interligados através de uma rede de computadores e equipados com *software* que permita o compartilhamento dos recursos do sistema: *hardware*, *software* e dados”. As principais características de sistemas distribuídos são: alto desempenho, compartilhamento de recursos, heterogeneidade, abertura, escalabili-

dade, tolerância a falhas, concorrência, e transparência.

Sistemas centralizados tendem a ter desempenho inferior a sistemas distribuídos. O desempenho é um conceito relativo que pode estar associado a diferentes grandezas como tempo de resposta do servidor, vazão², e quantidade de recursos consumidos pela rede (BARCELAR, 2013). O fato é que ao comparar tais grandezas testando dois sistemas, sendo o primeiro um sistema distribuído com algumas unidades de processamento, e o segundo centralizado com uma única unidade de processamento³, o sistema distribuído apresentará resultados superiores aos do sistema centralizado. Os sistemas distribuídos conseguem realizar uma tarefa em menos tempo quando a natureza do algoritmo é paralelizável, possibilitando a divisão do problema em sub-problemas que podem ser resolvidos por mais de um nó de processamento. Para que esta otimização seja eficiente, é preciso que o tempo levado para o compartilhamento de dados entre os nós do sistema seja compensado com o paralelismo de suas unidades de processamento (TANENBAUM; STEEN, 2007).

Uma das características mais básicas de sistemas distribuídos é o compartilhamento de recursos, uma vez que sem ela seria impossível conceber um sistema distribuído. O compartilhamento de recursos é tão comum no dia-a-dia de um usuário de computador e Internet que muitas vezes o mesmo não percebe que está fazendo uso de recursos compartilhados. Tal característica possibilita obter economia, quando se compartilha recursos de *hardware* como impressoras ou sistemas de armazenamento, além de facilitar a colaboração e troca de informações entre usuários separados geograficamente, utilizando recursos de *software*, como por exemplo, o correio eletrônico (COULOURIS, 2009).

Uma vez que sistemas distribuídos podem ser compostos de várias unidades independentes, existe a liberdade para que tais unidades tenham composição de *software* e *hardware* diferenciadas, para melhor atender a demanda de determinado sistema. Se por um lado a heterogeneidade possibilita a construção de um sistema distribuído a partir de várias redes, sistemas operacionais e linguagens de programação, por outro lado surge a necessidade de um componente que lide com essas diferenças, tornando-as transparentes: o *middleware*. O *middleware* consiste em uma camada de *software* com objetivo de prover abstração à heterogeneidade dos nós do sistema distribuídos, fornecendo um modelo computacional uniforme

²Número de requisições atendidas por unidade de tempo.

³Considerando que as unidades de processamento de ambos têm a mesma frequência.

para programadores (TANENBAUM; STEEN, 2007).

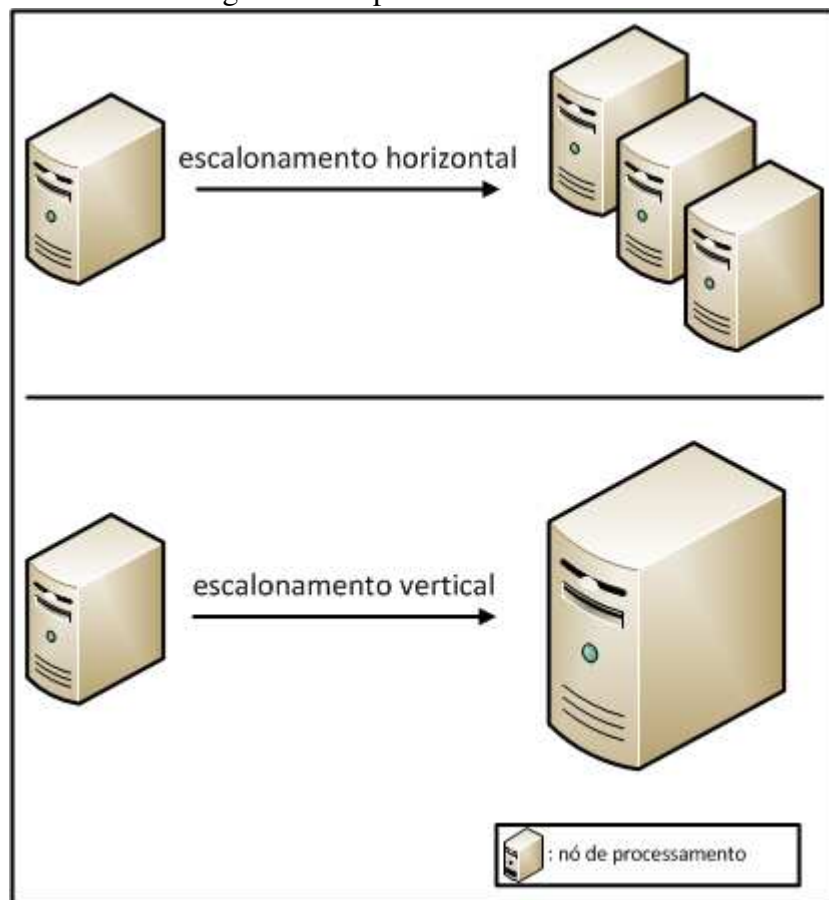
A abertura de um sistema distribuído é determinada pela capacidade que o mesmo tem de ser estendido e reimplementado de várias maneiras (COULOURIS; DOLLIMORE; KINDBERG, 2001). Para conceber um sistema distribuído com abertura é preciso que os serviços do mesmo sejam expostos publicamente por meio de uma interface e descritos por suas respectivas especificações, para que possam ser utilizados por terceiros. Adicionalmente, a oferta de seus serviços deve ser interoperável e portátil, permitindo que componentes de fornecedores diferentes coexistam, e que outros sistemas distribuídos sejam produzidos a partir de diferentes *hardware* e *software*. Em outras palavras, a abertura permite a heterogeneidade de sistemas distribuídos.

Sistemas distribuídos são denominados escaláveis quando conseguem prover um serviço de forma eficiente, principalmente quando submetidos a grandes cargas de requisições (COULOURIS; DOLLIMORE; KINDBERG, 2001). Segundo Barcelar (2013), uma das principais vantagens dos sistemas distribuídos em relação aos centralizados, é que “sistemas fortemente acoplados (centralizados) possuem limitação prática no aumento do desempenho, pois necessitam de uma infraestrutura especial de barramento e memória para aumentar o número de processadores”. Para suportar uma maior demanda, sistemas distribuídos são acrescidos de novos nós de processamento, sem necessidade de mudança adicional na arquitetura, uma vez que, geralmente, a mesma é projetada para suportar o acréscimo de novo *hardware* de maneira automatizada. Existem dois tipos difundidos de escalonamento (MICHAEL et al., 2007): o horizontal, e o vertical. Escalonar horizontalmente significa adicionar mais nós de processamento ao sistema. Em contrapartida, para escalonar verticalmente é preciso adicionar mais recursos (memória RAM⁴, núcleos de processamento, espaço para armazenamento, etc.) a um nó já pertencente ao sistema. A Figura 2.2 ilustra os dois tipos de escalonamento.

Sistemas computacionais também são suscetíveis a falhas. Quando falhas ocorrem, seja a nível de *software* ou *hardware*, os programas podem produzir resultado incorreto, ou simplesmente parar antes de completarem suas tarefas. Um sistema tolerante a falhas é aquele com capacidade de sobreviver perante a falha de algum de seus componentes (BARCELAR, 2013). Toda técnica de tolerância a falhas requer primeiramente a detecção da mesma, para

⁴Random Access Memory

Figura 2.2: Tipos de escalonamento



que posteriormente alguma ação seja efetuada. A técnica mais simples consiste em avisar ao cliente que determinada requisição não pôde ser processada pelo sistema, ou seja, aquela requisição não pode mais ser atendida, apesar do sistema continuar funcionando para processar novas requisições. Também é possível tolerar falhas através da redundância de componentes, elevando a confiabilidade⁵ do sistema (TANENBAUM; STEEN, 2007). Para tal, é preciso que haja mais de um componente disponível para realizar uma mesma função, pois se um falhar, as requisições podem ser redirecionadas para o outro. Falhas também podem ser toleradas a nível de *software*, sendo possível, por exemplo, retransmitir uma mensagem cuja transmissão foi problemática (COULOURIS; DOLLIMORE; KINDBERG, 2001).

Em sistemas distribuídos é normal que haja acesso a recursos compartilhados, o que resulta em concorrência. Um sistema distribuído, por exemplo, poderia ter uma fila que armazenaria as requisições dos clientes, e vários componentes que processariam esta fila.

⁵Capacidade do sistema de manter-se em funcionamento durante o maior intervalo de tempo possível.

Tais componentes, portanto, estariam realizando um acesso concorrente à fila para processar o primeiro elemento da mesma. É preciso, dessa forma, que o programador (ou *framework* utilizado) implemente estratégias para que o acesso a estes recursos sejam atômicos.

Transparência é um conceito chave para o desenvolvimento de sistemas distribuídos. Um sistema distribuído que é capaz de se apresentar a usuários e aplicações como se fosse um único sistema computacional é denominado transparente (TANENBAUM; STEEN, 2007). A função da transparência é abstrair a complexidade do sistema distribuído para programadores, de forma que eles se preocupem apenas com o projeto de seu programa em particular (BARCELAR, 2013). Um exemplo é que os programadores de aplicações ao utilizarem um serviço não precisam saber que os recursos de um sistema estão fisicamente distribuídos. A International Standards Organization (1992) identificou os principais tipos de transparência:

1. Transparência de acesso: oculta as diferenças na representação dos dados e no acesso a recursos locais e remotos;
2. Transparência de localização: abstrai a localização real do recurso;
3. Transparência de concorrência: permite que vários processos e/ou usuários utilizem o mesmo recurso de forma compartilhada sem interferência;
4. Transparência de replicação: múltiplas instâncias de um recurso podem ser utilizadas para aumentar a confiabilidade e desempenho sem que os usuários ou programadores de aplicação percebam;
5. Transparência de falhas: abstrai a ocorrência de falhas dos programadores de aplicações e usuários, permitindo que os mesmos completem suas tarefas;
6. Transparência de escalabilidade: permite que o sistema escale sem necessidade de alteração na estrutura do sistema ou algoritmo da aplicação.

A maioria das características de sistemas distribuídos supracitadas nesta seção integrarão o DAaaS. A medida em que a arquitetura do DAaaS for detalhada no Capítulo 4, serão descritos como cada uma destas características dessa beneficiará o sistema. Na próxima seção serão apresentados os principais conceitos e técnicas de computação em nuvem, que é um modelo e infraestrutura para implantação de sistemas distribuídos.

2.5 Computação em Nuvem

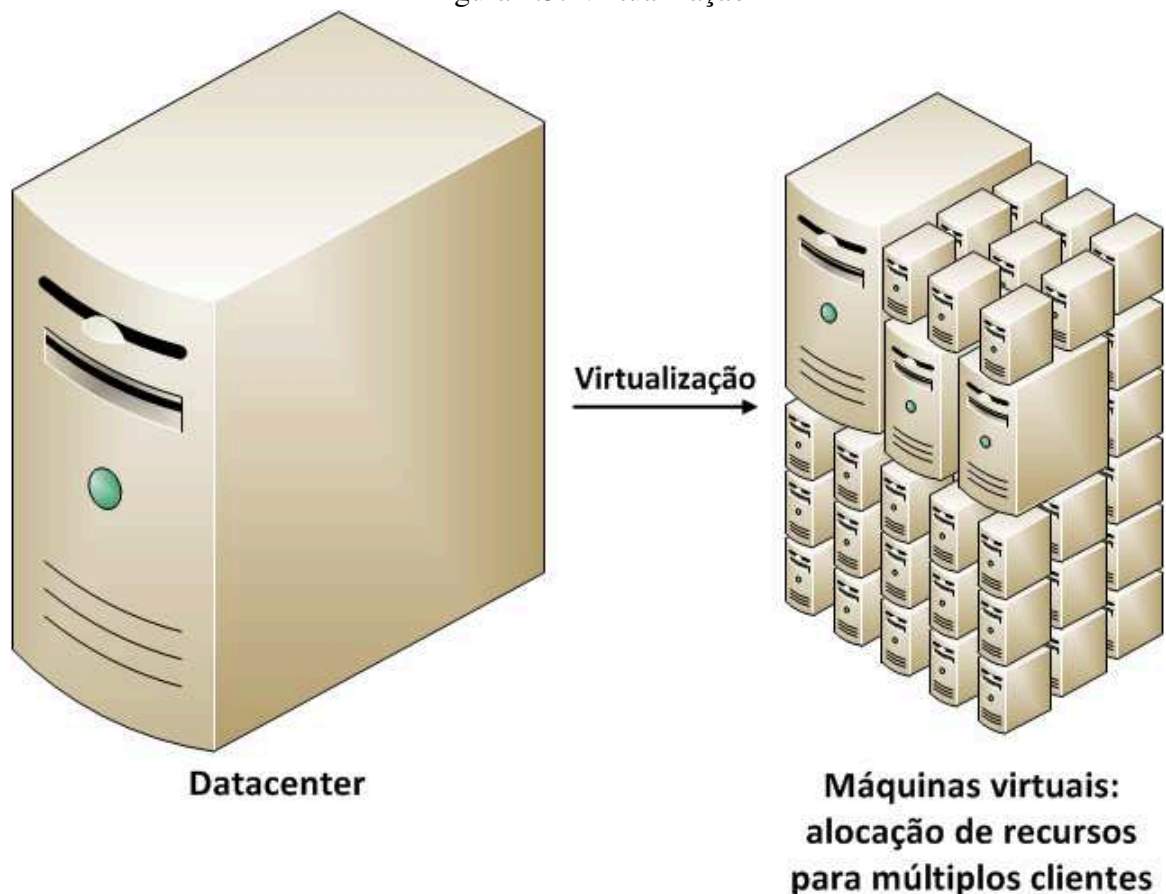
Em meados dos anos 70, cientistas começaram a vislumbrar a computação distribuída como um meio de superar as barreiras impostas pela limitada capacidade de *hardware*. É possível afirmar que a computação em nuvem surgiu primariamente da evolução de outros tipos de sistemas distribuídos, como a computação em grades e *clusters*. Contudo, é importante ressaltar que tal surgimento foi possibilitado pela evolução das Tecnologias de Informação (TI) e maturidade em outras áreas da computação, como por exemplo, avanços nas técnicas de virtualização alcançados nos últimos anos.

A computação está sendo transformada em um modelo baseado em serviços que são comoditizados e entregues de modo similar aos bens comuns como água, eletricidade, gás e telefonia (BUYYA et al., 2009). Este conceito é denominado *utility computing*, e significa que os usuários utilizam tais serviços de TI (armazenamento, processamento, rede, etc.) baseados exclusivamente em suas necessidades, pagando apenas pelo que usarem (ZHANG; CHENG; BOUTABA, 2010). Esta nova forma de prover serviço tem o potencial de transformar grande parte da indústria de TI, tornando *software* cada vez mais acessível aos usuários através de sua disponibilização como serviço. A computação em nuvem permite, por exemplo, que *startups* com ideias inovadoras não precisem de alto capital inicial para *hardware* e disponibilização de seus serviços, ou para investimento em recursos humanos para operá-los (ARMBRUST et al., 2010).

Buyya et al. (2009) se refere à nuvem como um sistema paralelo e distribuído consistindo em uma coleção de computadores inter-conectados e virtualizados que são provisionados dinamicamente e abstraídos como um ou mais recursos computacionais unificados. Um dos principais fatores que torna a computação em nuvem um paradigma atrativo é a economia. O que possibilita tal economia é o uso dos recursos computacionais em larga escala por múltiplos usuários, principalmente pelo uso da técnica de virtualização. A virtualização é uma técnica que abstrai os detalhes do *hardware* físico e provê recursos virtualizados para aplicações de alto nível (ZHANG; CHENG; BOUTABA, 2010). Geralmente, os provedores de infraestrutura de computação em nuvem utilizam *datacenters* com alto poder de processamento e os virtualizam, resultando em várias máquinas virtuais. A virtualização tem a capacidade de gerenciar os recursos computacionais dos *datacenters*, e de maneira dinâmica

atribuir e reatribuir tais recursos virtuais para que múltiplos clientes possam utilizá-los para aplicações sob demanda e de modo mais eficiente. No momento em que um cliente requisita um recurso, ele informa a capacidade de processamento, memória e armazenamento que deseja, podendo ainda utilizar o sistema operacional que mais se adeque a sua aplicação. A figura 2.3 ilustra o funcionamento da técnica de virtualização.

Figura 2.3: Virtualização

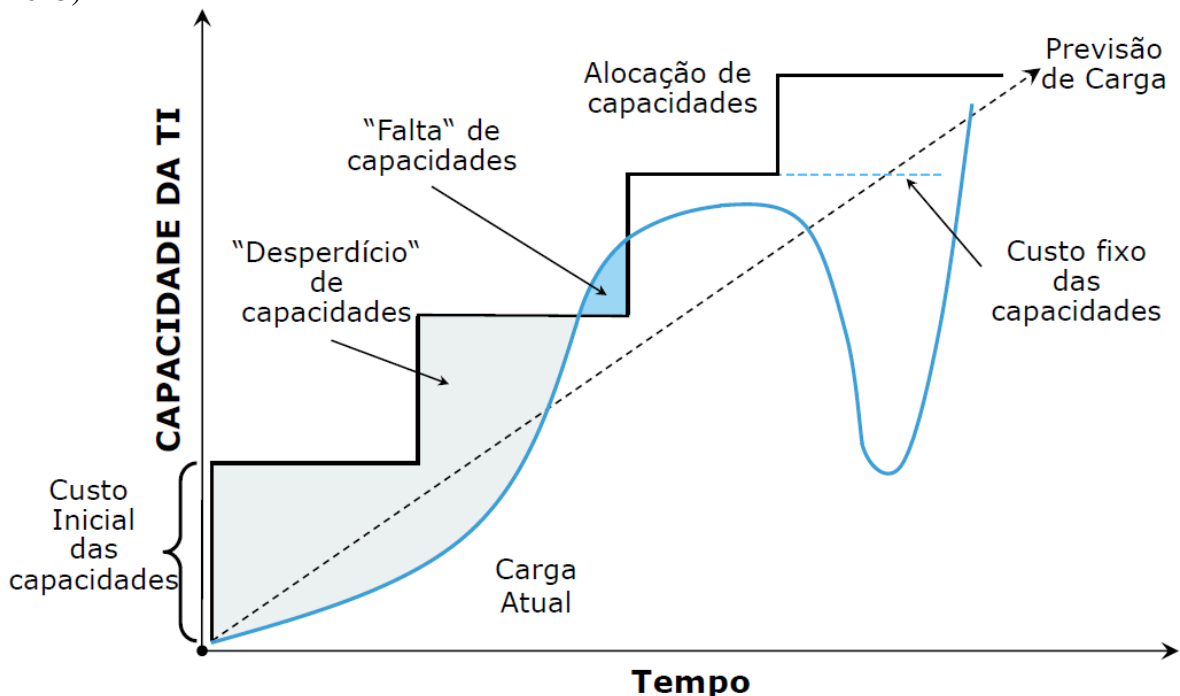


Uma das definições para computação em nuvem mais utilizadas por autores de livros e artigos é a do NIST (*National Institute of Standards and Technology*): computação em nuvem é um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (e.g., redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente adquiridos e liberados com mínimo esforço gerencial ou interação com o provedor de serviços (MELL; GRANCE, 2011). Mell e Grance (2011) ainda afirmam que a nuvem é composta por cinco características essenciais, três modelos de serviço, e quatro modelos de implantação.

2.5.1 Características Essenciais

A primeira característica essencial é o *self-service* sob demanda. Um consumidor pode provisionar recursos computacionais unilateralmente, como instâncias de processamento ou armazenamento por unidade de tempo, na medida em que for preciso, sem necessidade de interação humana com o provedor de serviço (MELL; GRANCE, 2011). Quando não se trabalha com o paradigma de computação em nuvem é normal que a empresa de TI adquira um *datacenter* baseado na previsão de carga que o serviço receberá. Analisando a Figura 2.4, é possível perceber que essa forma de utilização dos recursos tende a gerar subutilização do *hardware* e consequentemente gastos desnecessários, ou ainda, uma má oferta do serviço fornecido (ARMBRUST et al., 2010). Um motivo é que a quantidade de requisições pode ser muito acima da previsão realizada, gerando “falta de capacidades” na oferta do serviço, ou muito aquém, gerando “desperdícios de capacidades”. Porém, ao utilizar o paradigma de computação em nuvem, uma empresa é capaz de pagar pelo uso de recursos computacionais por hora, levando a uma redução potencial de custos.

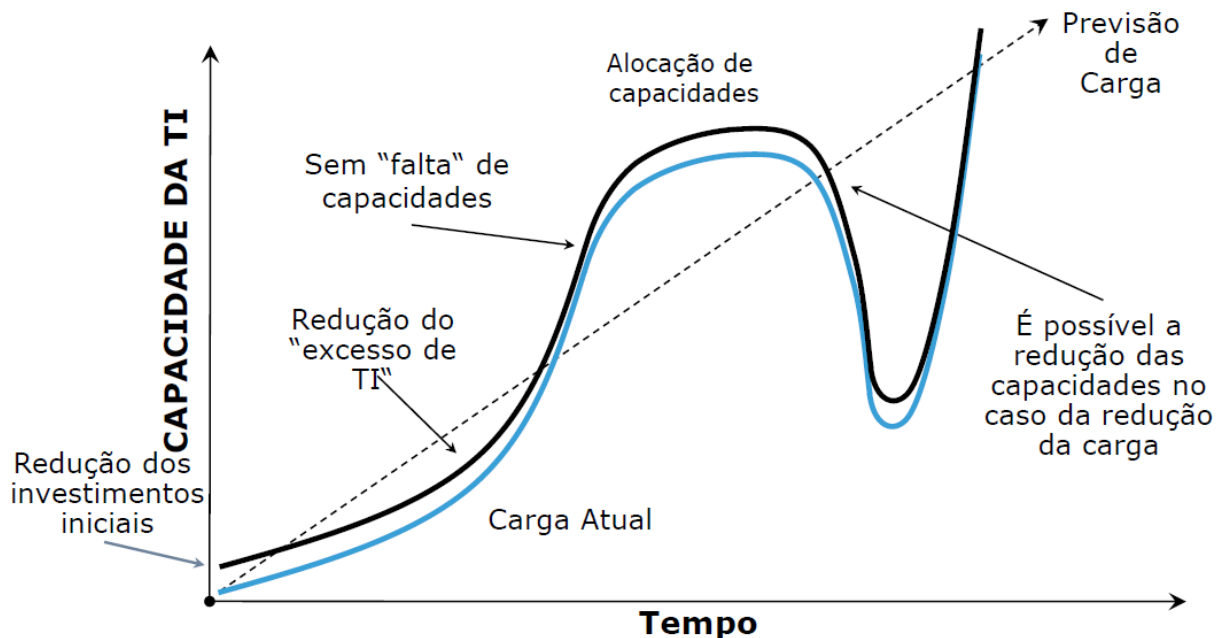
Figura 2.4: Empresa de TI que não utiliza o paradigma de computação em nuvem (SOUSA, 2013)



A segunda característica essencial é a rápida elasticidade. Recursos podem ser elasti-

camente provisionados e liberados, e em alguns casos de modo automático, para escalar rapidamente aumentando ou diminuindo suas capacidades de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento parecem ser ilimitados e podem ser provisionados a qualquer quantidade e momento (MELL; GRANCE, 2011). A capacidade de escalonamento rápido e dinâmico tem como objetivo resolver os problemas da alta variabilidade da demanda. Na Figura 2.4, é possível perceber que a demanda tem um alto nível de variabilidade, fazendo com que a infraestrutura de *hardware* de uma empresa de TI convencional seja subutilizada em determinados momentos, e em outros seja insuficiente para a demanda. Em contrapartida, se essa mesma empresa utilizar o paradigma de computação em nuvem, ela se beneficiará do provisionamento dinâmico e rápida escalabilidade através de uma infraestrutura elástica para conseguir atender a demanda com um serviço de qualidade e que não gere gastos desnecessários (ARMBRUST et al., 2010). A Figura 2.5 ilustra o oferecimento de recursos de *hardware* de maneira elástica e escalável, por uma empresa de TI que utiliza o paradigma de computação em nuvem.

Figura 2.5: Disponibilização de recursos de maneira elástica e escalável (SOUSA, 2013)



A terceira característica é o gerenciamento de recursos. Os recursos computacionais do provedor de infraestrutura são gerenciados para servir múltiplos clientes usando o modelo

*multi-tenant*⁶, com diferentes recursos físicos e virtuais dinamicamente atribuídos e reatribuídos de acordo com a demanda do cliente. Este serviço é fornecido de modo completamente transparente, ou seja, o cliente não tem informações sobre o local exato do recurso utilizado, podendo apenas requisitar uma localização em um nível mais alto (e.g., país, estado, ou *datacenter*) (MELL; GRANCE, 2011). Exemplos de recursos incluem armazenamento, processamento, memória, e banda de rede.

A quarta característica é o serviço medido. Sistemas na nuvem controlam e otimizam os recursos automaticamente provendo um nível de abstração apropriada ao tipo de serviço provido (MELL; GRANCE, 2011). O uso dos recursos são monitorados provendo transparência tanto para o provedor como consumidor do serviço utilizado. A computação em nuvem geralmente é ofertada com um modelo de preços que possibilita o cliente pagar pelo que usar⁷ e apenas pelos serviços que precisar. Por exemplo, se uma empresa precisar de 1000 instâncias computacionais por uma hora, ela pagará apenas por estas 1000 instâncias computacionais, e apenas pela hora que usá-las (GROSSMAN, 2009). Esse caso poderia ser interessante para empresas de TI que oferecem serviços de processamento em lotes, onde usar 1000 servidores por uma hora teria custo equivalente a usar um servidor por 1000 horas, oferecendo, portanto, serviços com mais agilidade e qualidade a seus usuários pelo mesmo preço (ARMBRUST et al., 2010).

A quinta característica é o acesso à rede de alta capacidade. Os recursos (sejam estes de *software* ou *hardware*) são disponibilizados através da rede e acessados por meio de mecanismos padrão que promovam seu uso por plataformas de cliente heterogêneas leves ou robustas (e.g., *smartphones*, *tablets*, *notebooks* e *workstations*) (MELL; GRANCE, 2011).

2.5.2 Modelos de Serviço

No paradigma de computação em nuvem tudo é disponibilizado em forma de serviço, desde o nível de aplicação até o de infraestrutura. Para tal, faz-se necessário uma forte ênfase na gestão de serviços. Na nuvem, cada provedor de serviços (a nível de *software* ou *hardware*) oferece seus serviços respeitando um “acordo de nível de serviço” (do inglês *Service Level Agreement* - SLA) negociado com seus clientes (ZHANG; CHENG; BOUTABA, 2010). O

⁶Compartilhamento de recursos físicos para vários clientes através de técnicas de virtualização.

⁷“Pay as you go”: termo comumente utilizado para expressar as características do serviço medido e *self-service* sob demanda.

SLA especifica todas as prioridades, responsabilidades, e garantias que a empresa provedora de serviços oferece a seus clientes, transmitindo maior segurança para os mesmos.

Mell e Grance (2011) afirmaram que a computação em nuvem é baseada prioritariamente na entrega de três modelos de serviço: *software* como um serviço (do inglês *Software as a Service* - SaaS), plataforma como um serviço (do inglês *Platform as a Service* - PaaS), e infraestrutura como um serviço (do inglês *Infrastructure as a Service* - IaaS). O IaaS é a base de toda computação em nuvem e provê suporte ao PaaS que por sua vez provê suporte ao SaaS, podendo ainda fornecer suporte direto ao SaaS sem intermédio do PaaS. O IaaS é o modelo de serviço com o menor nível de abstração, ou seja, o consumidor consegue controlar em um dado nível a quantidade de instâncias de processamento que deseja utilizar, ao passo que no SaaS isso fica completamente transparente para o usuário final. Esses três modelos de oferta de serviço são os principais, e várias outras empresas baseiam-se neles para ofertar os mais variados serviços (e.g., *Salesforce*, *Amazon*, *Amazon Web Services*, etc.). Este modelo de negócio baseado em serviço se tornou extremamente comum no mercado atual e criou uma nova tendência denominada de “XaaS”, onde X representa os mais diversos tipos de serviços capazes de serem oferecidos pela computação em nuvem. Um exemplo é o conceito do atual trabalho, DAaaS, que visa prover acessibilidade para surdos como um serviço através da nuvem. A Figura 2.6 exemplifica como os modelos de serviço interagem entre si.

Infraestrutura como um Serviço

Infraestrutura como um serviço é a capacidade de prover ao cliente recursos computacionais tais como processamento, armazenamento, rede, etc., permitindo a este cliente implantar e rodar um *software* arbitrário (seus serviços) (MELL; GRANCE, 2011). O provedor de infraestrutura dispõe de uma camada de *software* e *hardware* de baixo nível capazes de provisionar ou liberar recursos para seus clientes. Desse modo, o cliente não gerencia ou tem acesso direto às camadas inferiores da nuvem (infraestrutura de *hardware*) mas tem controle sobre os sistemas operacionais, armazenamento, e aplicações que deseja usar.

A aquisição, instalação, e manutenção de *datacenters* é uma atividade trabalhosa e que gera muitos gastos. Um dos principais benefícios da computação em nuvem é a economia. Empresas provedoras de IaaS obtém lucros pelo fato de através da virtualização buscar ex-

Figura 2.6: Modelos de serviço (SOUSA, 2013)



plorar ao máximo seu *hardware* quando os disponibilizam a seus clientes. Empresas que utilizam a infraestrutura como um serviço (empresas que provêem SaaS) em larga escala passam a obter economia proveniente do fato de não gastarem com eletricidade, rede, manutenção operacional, e *hardware*, onde este último requer alto capital inicial que nem sempre tais empresas dispõem (ARMBRUST et al., 2010). Um exemplo de empresa que provê IaaS é a *Amazon Web Services* através de serviços como o *Elastic Compute Cloud* (ZHANG; CHENG; BOUTABA, 2010).

Plataforma como um Serviço

Quando uma empresa fornece plataforma como um serviço, seu objetivo é prover recursos para implantação dos serviços do cliente na nuvem. Linguagens de programação, bibliotecas, serviços e outra ferramentas são oferecidos pelo provedor de PaaS para que o cliente possa focar no desenvolvimento de sua aplicação. Adicionalmente, usuários de PaaS não precisam gerenciar ou controlar as camadas inferiores da infraestrutura da nuvem como rede, servidores, sistemas operacionais, ou armazenamento, uma vez que estas são fornecidas

e gerenciadas pela IaaS (MELL; GRANCE, 2011). Exemplos de empresas que proveem PaaS são *Google App Engine*, *Microsoft Windows Azure*, e *Amazon Web Services* através de serviços como o *Elastic MapReduce* (ZHANG; CHENG; BOUTABA, 2010).

Software como um Serviço

Software como um serviço é o modelo mais visível ao usuário final. No SaaS os usuários utilizam as aplicações que são executadas diretamente na infraestrutura de nuvem. Este modelo de serviço é completamente transparente para o usuário final, isto é, o mesmo não tem acesso às camadas inferiores de infraestrutura da nuvem (rede, servidores, sistemas operacionais, armazenamento), podendo ter acesso apenas às configurações de usuário para a aplicação específica (MELL; GRANCE, 2011). Tais aplicações são acessíveis por vários meios como navegadores de Internet, interfaces gráficas de programas *standalones*, ou até mesmo *smartphones* e *tablets*. O processamento destas aplicações é completamente transferido aos *datacenters* onde as aplicações foram implantadas. Tal fato diminui as restrições de *hardware* necessárias ao usuário final, além de otimizar o desempenho das aplicações que antes executavam no computador do usuário (com capacidade de *hardware* mais limitada) sem a necessidade de investimento por parte do mesmo (PIGATTO, 2009). Adicionalmente, o usuário final ganha em usabilidade, uma vez que se abstém da instalação, atualização, ou manutenção destes programas em seu computador. Essas atividades passam a ser realizadas exclusivamente pelo provedor do serviço diretamente na nuvem. Um exemplo de *software* disponibilizado como serviço é o *Google Docs*. Com este aplicativo o usuário final pode editar texto sem a necessidade de instalação ou manutenção do *software*.

2.5.3 Modelos de Implantação

Mell e Grance (2011) afirmam que existem quatro modelos para a implantação de uma nuvem: público, privado, híbrido e comunitário. O gerenciamento, custo, e segurança das nuvens depende primariamente da escolha da organização em comprar e operar sua própria nuvem ou obter serviços em nuvem de forma terceirizada (GROSSMAN, 2009).

No modelo público, a infraestrutura de nuvem é aberta ao público geral que por sua vez utilizam recursos compartilhados na nuvem. Nuvens públicas oferecem uma gama de

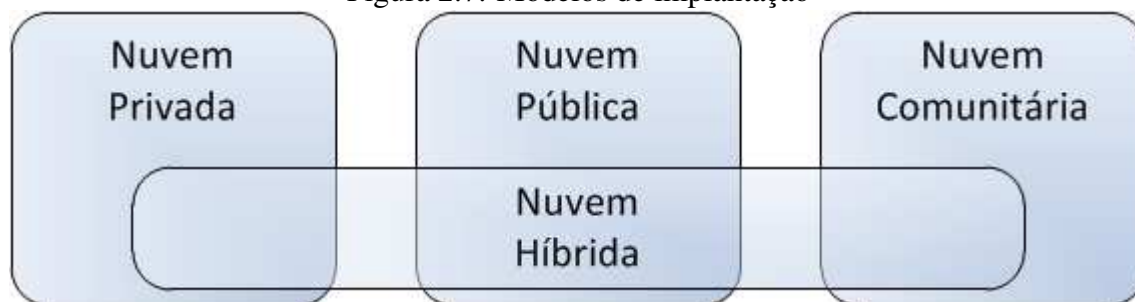
benefícios a seus clientes, que vão desde a falta de necessidade de investimento inicial em infraestrutura à não preocupação com os riscos naturais que o *hardware* apresenta. Todavia, nuvens públicas não permitem que seus clientes tenham controle minucioso dos dados, rede e configurações de segurança, o que dificulta sua eficácia em alguns cenários de negócio (ZHANG; CHENG; BOUTABA, 2010). Um exemplo de nuvem pública são os serviços fornecidos pela *Amazon: Elastic Compute Cloud, Simple Storage Service, SimpleDB*, etc.

No modelo privado, a nuvem pertence exclusivamente a uma organização, tendo como usuários os vários departamentos da mesma (MELL; GRANCE, 2011). É normal utilizar nuvens privadas quando a empresa é grande o suficiente para beneficiar-se das vantagens econômicas da computação em nuvem comentadas previamente (ARMBRUST et al., 2010). A nuvem privada pode ser gerenciada pela própria organização, por uma terceira parte, ou por alguma combinação de ambas. O *Google*, por exemplo, utilizava o *Google File System*, *MapReduce* e *BigTable* internamente como parte de seus serviços de nuvem privados, e naquele determinando momento (2009), tais serviços não eram disponibilizados a terceiros (GROSSMAN, 2009).

No modelo de nuvem comunitária, a infraestrutura da nuvem é provisionada exclusivamente para uso de uma comunidade específica de clientes de organizações que tem requisitos comuns (e.g., a missão, requisitos de segurança, política e considerações de conformidade) (MELL; GRANCE, 2011).

No modelo de nuvem híbrido, a nuvem pode ser uma combinação de nuvens públicas e privadas (ou comunitárias). Esse modelo de implantação oferece mais flexibilidade do que as nuvens exclusivamente públicas ou privadas, pois parte da nuvem pode ser privada e oferecer controle e segurança mais rígidos sobre os dados da aplicação, e a outra parte pode ser pública para facilitar o escalonamento em grandes quantidades para serviços sob demanda (ZHANG; CHENG; BOUTABA, 2010). A Figura 2.7 ilustra os diferentes modelos de implantação na nuvem.

Figura 2.7: Modelos de implantação



2.5.4 Balanceamento de Carga e Provisionamento Dinâmico de Recursos

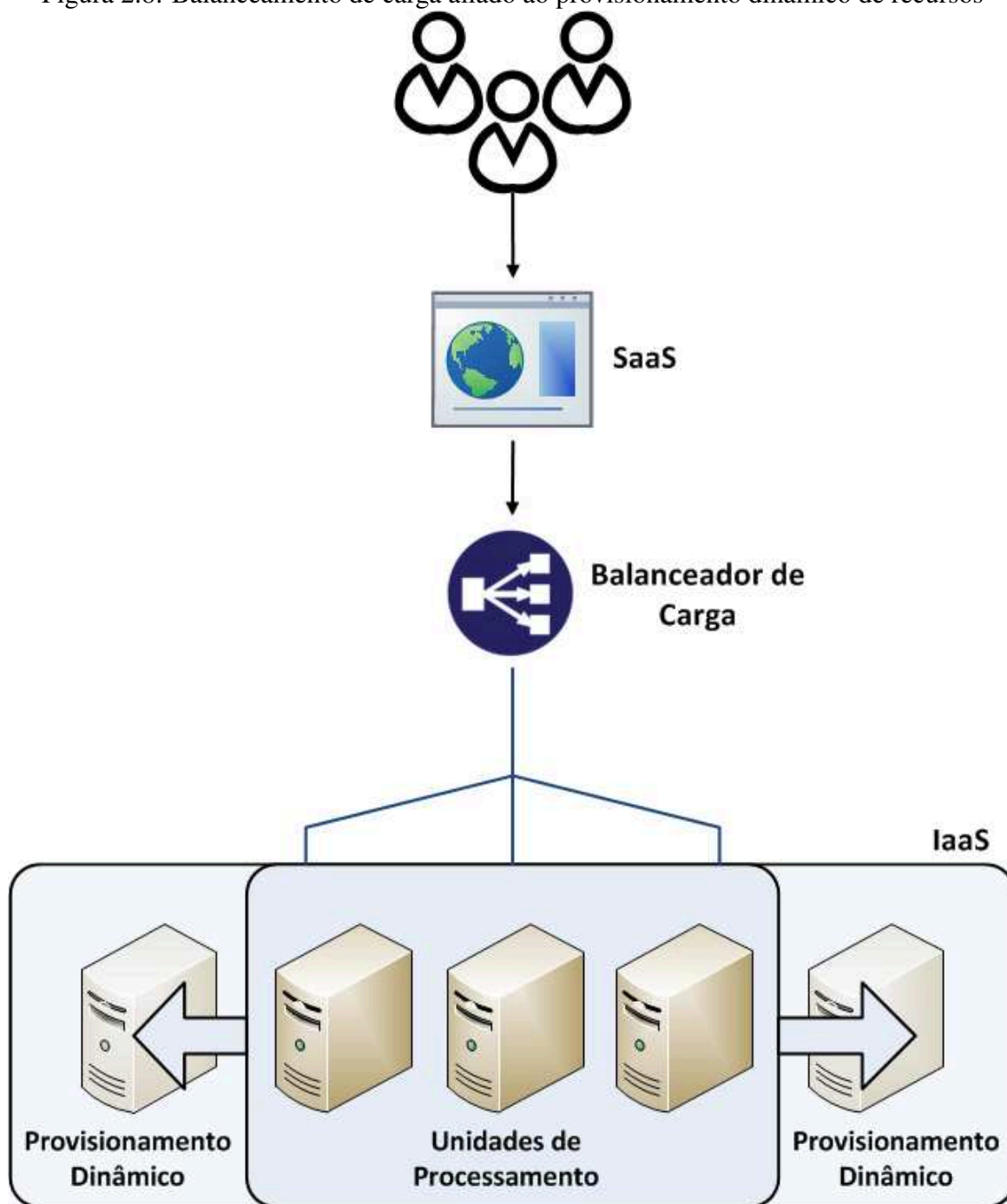
Um dos objetivos mais comuns de clientes que utilizam a nuvem é atender a demanda variável de seu sistema de forma eficiente, com agilidade e qualidade. Para isso, duas técnicas de computação em nuvem bem difundidas são utilizadas: balanceamento de carga, e provisionamento dinâmico de recursos.

Quando uma empresa disponibiliza seus serviços publicamente sem utilizar o paradigma de computação em nuvem, geralmente ela busca adquirir um *datacenter* que tenha poder de processamento suficiente para atender a previsão de sua demanda. Entretanto, em alguns momentos podem existir picos de demandas no qual a infraestrutura da empresa já não suportaria mais. Na nuvem, é possível utilizar o provisionamento dinâmico para requisitar de maneira automática mais recursos como armazenamento, processamento, memória RAM, largura de banda de rede, ou até novas unidades de processamento à organização que provê IaaS. Adicionalmente, é preciso utilizar o balanceamento de carga para que essas novas instâncias de processamento possam receber parte do fluxo de requisições, dividindo o trabalho total para várias unidades de processamento. A Figura 2.8 ilustra o funcionamento desta técnica.

2.5.5 Amazon Web Services

Grande parte da nuvem é baseada em provedores de IaaS. Cada provedor possui detalhes que o diferenciam dos demais, como por exemplo, o custo e as limitações impostas. No presente trabalho, o provedor de IaaS utilizado é a *Amazon Web Services*, uma vez que a

Figura 2.8: Balanceamento de carga aliado ao provisionamento dinâmico de recursos



empresa forneceu bonificação de uso de seus serviços para a presente pesquisa científica. Contudo, a arquitetura proposta para o presente trabalho é genérica, podendo ser implantada em qualquer outro provedor de IaaS.

A AWS oferece um conjunto de serviços de aplicativos e infraestrutura de computação em nuvem para atender seus clientes nos mais variados nichos específicos, desde a necessi-

dade de altas capacidades de processamento até a alta demanda de disponibilidade. Uma das principais vantagens da computação em nuvem é a oportunidade de substituir diretamente os gastos com a infraestrutura principal por variados preços baixos, que se ajustam de acordo com o cliente.

Amazon Elastic Compute Cloud (Ec2)

O *Amazon Elastic Compute Cloud* é um serviço que fornece uma capacidade de computação redimensionável na nuvem (AWS, 2013a). Com este serviço, o cliente é capaz de criar novas instâncias (unidades de processamento), que na verdade são computadores com configuração flexíveis, sendo possível escolher o sistema operacional além de capacidades de processamento, memória RAM e armazenamento. O *Amazon Ec2* reduz para alguns minutos o tempo exigido para obter e inicializar novas instâncias do servidor, possibilitando o cliente instalar e configurar remotamente as aplicações necessárias para tornar seu serviço disponível na instância. Portanto, é possível escalonar a capacidade de processamento para mais ou menos em minutos.

Amazon Elastic Load Balancing (ELB)

O *Amazon Elastic Load Balancing* é um balanceador de carga (como descrito em 2.5.4) que tem capacidade de distribuir automaticamente o tráfego de entrada dos aplicativos em várias instâncias do *Amazon Ec2* (AWS, 2013f). Com ele é possível tornar os serviços mais tolerantes a falhas, distribuindo automaticamente o tráfego de entrada para as instâncias íntegras até que as instâncias com problemas sejam restauradas. Outra maneira de prevenção contra falhas é através da alta disponibilidade dos serviços. Os clientes podem habilitar o ELB dentro de uma única zona de disponibilidade ou em várias para alcançar um desempenho de aplicativo ainda mais consistente, desse modo, se houver algum problema em uma zona de disponibilidade (como falta de energia, problema com a rede, etc.) o tráfego será direcionado para as zonas que estiverem em pleno funcionamento.

Quando se utiliza os serviços da AWS é possível escolher em alto nível a localização de seus recursos através de regiões e zonas de disponibilidade. As regiões são o nível mais abrangente de localização, permitindo o cliente escolher em qual país e/ou estado deseja utilizar os recursos. A AWS disponibiliza oito regiões: *US East (N. Virginia)*, *US West (Ore-*

gon), *US West (N. California)*, *EU (Ireland)*, *Asia Pacific (Singapore)*, *Asia Pacific (Tokyo)*, *Asia Pacific (Sydney)*, *South America (São Paulo)* (AWS, 2013g). Zonas de disponibilidade são posições distintas (*datacenters*) dentro das regiões, e são projetadas para serem independentes.

Auto Scaling

O *Auto Scaling* é um serviço fornecido pela AWS que permite escalonar automaticamente a capacidade das instâncias Ec2 de acordo com condições pré-definidas pelo cliente (AWS, 2013e). Desse modo, é possível configurar o *Auto Scaling* para aumentar o número de instâncias em picos de demanda para manter um bom desempenho, mas também é possível diminuir essa quantidade durante quedas de demanda para minimizar os custos. Para o funcionamento do *Auto Scaling* é preciso configurar gatilhos e políticas de escalonamento. Os gatilhos definirão quando escalonar e as políticas irão definir como escalonar. Um exemplo seria configurar um gatilho para ser disparado quando as instâncias Ec2 atingissem 90% de processamento, e a política de escalonamento para criar uma nova instância (com o serviço pré-configurado). De modo geral, uma nova instância seria criada para cada vez que uma instância chegasse a 90% de processamento.

O *Auto Scaling* é um serviço que funciona em conjunto com o ELB, pois cada nova instância criada só recebe demanda graças ao balanceamento de carga. Também é necessário utilizar o *Amazon Machine Image* (AMI), para que cada nova instância criada inicie com a imagem dos serviços do cliente já configurados e prontos para receber novas requisições.

Amazon Simple Queue Service (SQS)

Em sistemas distribuídos é fundamental que as unidades de processamento possam trocar informações para que o sistema funcione de modo colaborativo. A *Amazon* fornece o SQS como meio simples para troca de mensagens por meio de filas. O SQS se encaixa bem em sistemas que tenham componentes no estilo produtor-consumidor, podendo ter um ou vários componentes produtores e/ou consumidores. Eventualmente, um serviço pode ficar sobrecarregado, mas é possível tratar essa sobrecarga mantendo as requisições em uma fila, sem necessidade de escalonamento das instâncias Ec2 (se pertinente), e processar tais requisições na medida em que o sistema tenha baixa demanda.

As filas SQS armazenam as mensagens de maneira duradoura até que o servidor as processe. Quando as mensagens são lidas elas não são automaticamente removidas, mas são configuradas como invisível, para que nenhum outro componente a processe repetidamente (AWS, 2013b). A invisibilidade é definida por um *visibility timeout* que por padrão tem valor 30 segundos. Uma vez processadas as mensagens, é necessário removê-las explicitamente da fila SQS, caso contrário, as mensagens tornar-se-ão novamente disponíveis na fila para que outros componentes a processem.

Amazon Simple Storage Service (S3)

O S3 é um serviço em nuvem para armazenamento escalável de arquivos. Na medida em que o cliente precisar de mais espaço, a *Amazon* por meio do S3 o fornece de maneira elástica e automática, tornando esse processo transparente ao cliente. O S3 é o armazenamento que foi projetado para facilitar a computação em larga escala na *Web*, preparado para serviços que realizem muitas requisições simultâneas, e fornece desde 1 *byte* até 5 *terabytes* (AWS, 2013c).

Amazon DynamoDB

O *Amazon DynamoDB* é um serviço que provê as principais funções de um banco de dados voltado para as nuvens. O objetivo principal dele é prover transparência aos desenvolvedores. Com ele, os desenvolvedores simplesmente armazenam e consultam itens de dados por meio de solicitações e o *Amazon DynamoDB* simplesmente escala quando necessário, abstraindo essa funcionalidade (AWS, 2013d). Ele é um banco de dados não relacional, e, portanto, trabalha com uma abordagem que elimina a necessidade de modelagem de dados, constantes manutenções e preocupação com aumento de desempenho, o que normalmente acontece com desenvolvedores que utilizam bancos relacionais (PIGATTO, 2009). O serviço oferece um conjunto de APIs para armazenamento e acesso aos dados, pagando somente por recursos utilizados.

2.6 Considerações

Nesse Capítulo foram apresentados os principais conceitos que fundamentam este trabalho. Foram explicadas as principais características de línguas de sinais e alguns detalhes da gramática de LIBRAS, exemplificando como algumas sentenças da língua portuguesa seriam representadas em LIBRAS. Posteriormente, foi detalhado o funcionamento geral da arquitetura do VLIBRAS, uma vez que o presente trabalho se baseia neste sistema de tradução automática. Por fim, foram apresentados os principais conceitos de sistemas distribuídos e computação em nuvem, com objetivo de transmitir ao leitor as vantagens que um serviço fornecido na nuvem tem quando comparado a um sistema centralizado. São essas vantagens proporcionadas pela computação em nuvem que se pretende incorporar no presente trabalho: transparência, flexibilidade para escalabilidade além de alta disponibilidade, tolerância a falhas, uso eficiente de recursos e pagamentos destes apenas perante o uso.

No próximo Capítulo serão apresentados os principais trabalhos relacionados presentes na literatura.

Capítulo 3

Trabalhos Relacionados

Nesse Capítulo serão apresentados os principais trabalhos relacionados presentes na literatura, categorizados em dois sub-grupos:

1. trabalhos relacionados à disponibilização de acessibilidade para surdos brasileiros nas TICs;
2. trabalhos que abordem técnicas de computação em nuvem para oferecer escalabilidade e tolerância a falhas, e sistemas de processamento multimídia implantados na nuvem.

Essa sub-divisão é decorrente das duas principais motivações do trabalho: oferecer uma API, para desenvolvedores de *software*, que proveja acessibilidade para surdos em forma de serviço, e validar uma proposta de arquitetura escalável e tolerante a falhas para um sistema de processamento multimídia.

3.1 Acessibilidade para Surdos Brasileiros nas TICs

Um dos objetivos do presente trabalho é permitir que desenvolvedores de sítios, aplicativos e afins, possam tornar seus conteúdos acessíveis de forma transparente, ou seja, sem preocupar-se em como ocorre o processo de tradução. Portanto, é interessante identificar os trabalhos que forneçam uma API/serviço a outros desenvolvedores de *software*, para facilitar a inclusão de conteúdos em seus programas de forma acessível aos surdos do Brasil.

Utilizando como palavras de busca “*accessibility for deaf people*” e “acessibilidade para surdos” no *Google* acadêmico e portal de periódicos CAPES foram encontrados muitos tra-

balhos que provêem acessibilidade a pessoas com deficiência auditiva, mas nenhum deles expõe seus serviços por meio de uma API pública a outros desenvolvedores. Contudo, a fim de situar a relevância do presente trabalho, esta seção abordará os trabalhos mais relevantes que tratem da concepção de ferramentas que disponibilizem acessibilidade para surdos brasileiros nas TICs, i.e., trabalhos que realizem a tradução automática da língua portuguesa para LIBRAS. Deste modo, será utilizado o trabalho “Tradutores Automáticos da Linguagem Português Oral e Escrita para uma Linguagem Visual-Espacial da Língua Brasileira de Sinais” como principal referência, uma vez que o mesmo realizou uma revisão sistemática, buscando resumir as principais evidências sobre tradutores automáticos da língua portuguesa para LIBRAS (PIVETTA; ULBRICHT; SAVI, 2011).

3.1.1 Rybená

O projeto Rybená (ICTS, 2013) (AMORIM et al., 2010) (MOREIRA et al., 2011) é uma solução que realiza tradução automática de conteúdos textuais em língua portuguesa para LIBRAS. A ferramenta é multiplataforma e tem diferentes versões para dispositivos móveis e Internet (MOREIRA et al., 2011), ou até mesmo TV (AMORIM et al., 2010). Na versão *Web*, o *Player* Rybená funciona como um *plugin* capaz de realizar a tradução de textos em língua portuguesa, como o conteúdo de qualquer página da Internet, para a LIBRAS. A versão *mobile* (Torpedo Rybená) presta um serviço que permite receber e enviar mensagens de texto traduzido para a LIBRAS, possibilitando a comunicação ouvinte -> surdo e surdo -> surdo, através de dispositivos móveis (PIVETTA; ULBRICHT; SAVI, 2011). Um ponto negativo é que o conteúdo em língua portuguesa é traduzido para português sinalizado, ou seja, ao invés do tradutor gerar a visualização dos sinais a partir da glosa para aquele texto, ele os gera a partir do texto na língua portuguesa. O *plugin* para a Web é proprietário e o custo fica em torno de R\$ 4.000,00 (quatro mil reais) (PIVETTA; ULBRICHT; SAVI, 2011). Em sua versão atual, a apresentação dos sinais em LIBRAS é produzida a partir de uma animação de um *avatar* 3D.

3.1.2 F-LIBRAS

Baptista (2007) desenvolveu a ferramenta F-LIBRAS, cuja principal funcionalidade é a tradução da língua portuguesa para LIBRAS. Adicionalmente, o *software* permite a criação e inclusão de novos sinais no dicionário, através da configuração visual de um *avatar* em um ambiente virtual 3D. Tal módulo tem função semelhante a do componente WikiLIBRAS, presente no tradutor VLIBRAS. F-LIBRAS é um programa *standalone*, produzido com a linguagem de programação Java e API Java 3D.

3.1.3 FALIBRAS

Coradine et al. (2003) propuseram o sistema FALIBRAS, com o intuito de apoiar a comunicação oral entre pessoas ouvintes e surdas do Brasil. O FALIBRAS recebe como entrada um áudio, que pode ser, por exemplo, de um professor com um microfone em uma sala de aula, e através da tecnologia para reconhecimento de voz *Viavoice* a converte para um texto equivalente. A próxima etapa é analisar o texto captado usando um interpretador léxico-morfológico-sintático, para a partir da classificação reorganizar a ordem dos sinais na frase, e, posteriormente, animá-los em tempo real a partir de um dicionário de gráficos vetoriais. A primeira versão do FALIBRAS foi disponibilizada para a tradução de orações simples para dispositivos móveis ou plataforma cliente-servidor. Portanto, um ponto negativo é sua ineficiência para tradução de textos longos. Um ponto positivo é a extensão que Franco e Brito (2011) desenvolveram, a partir da integração do projeto FALIBRAS ao navegador *Web Firefox*, gerando o *plugin* FALIBRAS-WEB. Esta extensão pode ser considerada um ponto positivo uma vez que viabiliza a acessibilidade de conteúdos de sítios para surdos brasileiros.

3.1.4 TLIBRAS

Para realizar a tradução da língua portuguesa para LIBRAS, o projeto TLIBRAS (LIRA, 2003) consiste de duas etapas:

1. Tradução em tempo real do português para LIBRAS, a partir de uma entrada via texto ou voz;
2. Apresentação da animação resultante da tradução em salas de aula, TV Digital ou

vídeos na internet (LIRA, 2009).

Os sinais são reproduzidos por um *avatar* 3D que recebe as coordenadas para posição de mãos e braços. Para tanto, a ferramenta contém um dicionário de sinais em LIBRAS, cujo conteúdo é o mapeamento dos membros do *avatar*.

3.1.5 VE-LIBRAS

VE-LIBRAS ou Veris é um projeto desenvolvido em 2009 com base no Rybená. Assim como o TLIBRAS e o FALIBRAS, o VE-LIBRAS aceita a voz humana como entrada. O VE-LIBRAS implementou uma camada de abstração superior ao Rybená, que recebe um áudio como entrada e o converte para texto (PIVETTA; ULBRICHT; SAVI, 2011). Posteriormente, o texto obtido do áudio é submetido ao Rybená, que está em uma camada inferior, e funciona como uma “caixa preta”, retornando como resultado a animação em LIBRAS.

3.1.6 POLI-LIBRAS

Segundo o sítio do POLI-LIBRAS (JANUÁRIO; KOGA; LEITE, 2013), Poli-Libras é “um conjunto de projetos *open-source* voltados para a comunidade surda, falante de LIBRAS, sendo seu principal objetivo a criação de um tradutor Português-Libras”. A ferramenta POLI-LIBRAS traduz frases em português para uma sequência de sinais em LIBRAS, de modo automatizado. Assim como o TLIBRAS, FALIBRAS, e F-LIBRAS, o processo de tradução do POLI-LIBRAS leva em conta aspectos sintáticos das línguas envolvidas, evitando a geração do chamado português sinalizado (JANUÁRIO; KOGA; LEITE, 2010). O POLI-LIBRAS também tem um componente chamado Wiki-Libras que permite a inclusão de novos sinais ao dicionário. Além do Wiki-Libras, outros pontos positivos são o fato de seu código ser aberto, e sua disponibilização ser através de um *plugin* para navegadores de Internet.

3.1.7 ProDeaf

Proativa (2013) explica o ProDeaf como “um conjunto de soluções de *software* capazes de traduzir texto e voz de português para LIBRAS com o objetivo de permitir a comunicação

entre surdos e ouvintes”. Assim como o TLIBRAS, FALIBRAS, e VE-LIBRAS, O ProDeaf é capaz de traduzir por meio de reconhecimento de voz. O projeto ProDeaf possui diferentes versões que dão suporte à plataforma *Web* e móvel. Podem ser citados como pontos positivos a gratuidade da versão para dispositivos móveis (plataforma *Android*, *iOS* e *Windows Phone*) e a tradução entre língua portuguesa e LIBRAS levando em conta suas diferenças gramaticais.

3.1.8 VLIBRAS

O sistema VLIBRAS é composto por um conjunto de componentes responsáveis por gerar automaticamente a tradução em LIBRAS. O VLIBRAS possui arquitetura flexível e realiza a tradução para diferentes plataformas: *Web* (ARAÚJO, 2012), PC e TV Digital (FERREIRA, 2012). O sistema aceita as seguintes entradas: texto, áudio, vídeo ou legenda. Para auxiliar na expansão do dicionário de sinais, o VLIBRAS possui um componente chamado WikiLIBRAS que permite que especialistas em LIBRAS gerem novos sinais, e insiram novas regras de tradução. O componente de tradução automática do VLIBRAS leva em consideração as diferenças gramaticais da língua portuguesa e LIBRAS, e, portanto, mapeia o conteúdo para uma sequência de glosas em LIBRAS. Apesar do VLIBRAS não ser *open-source*, seu código foi disponibilizado para o presente trabalho com fins de pesquisa.

3.1.9 Síntese

Diante dos sistemas de tradução português -> LIBRAS pesquisados, é interessante comparar as principais características dos mesmos, com objetivo de justificar a escolha do VLIBRAS como núcleo da API DAaaS. As características avaliadas são:

1. formatos de entrada aceitos: texto, áudio, vídeo, legenda;
2. plataformas de execução: PC, *Web*, *mobile*, televisão digital;
3. forma de tradução: se a representação visual dos sinais é feita utilizando português sinalizado, ou utilizando a glosa (representação textual em LIBRAS);
4. Dicionário expansível: se o projeto disponibiliza alguma ferramenta que permita a expansão do dicionário de sinais por especialistas em LIBRAS, e.g., o WikiLIBRAS.

A tabela 3.1 lista os trabalhos previamente mencionados com suas respectivas características.

Tabela 3.1: Sistemas de Tradução Automática Português -> LIBRAS. Adaptado de: (PI-VETTA; ULBRICHT; SAVI, 2011).

Tradutor	Entradas ¹				Plataformas ²				Tradução	Código	Dic. expansível
	T	A	V	L	PC	W	M	TV			
F-LIBRAS	X				X				glosa	fechado	X
FALIBRAS	X	X			X	X	X		glosa	fechado	
POLI-LIBRAS	X				X	X			glosa	aberto	X
ProDeaf	X	X				X	X		glosa	fechado	
Rybená	X					X	X	X	port. sinalizado	fechado	
TLIBRAS	X	X			X			X	glosa	fechado	
VLIBRAS	X	X	X	X	X	X		X	glosa	fechado ³	X
VE-LIBRAS	X	X			X				port. sinalizado	fechado	

A característica mais importante a ser avaliada é a forma de tradução. Os trabalhos que convertem o texto em língua portuguesa para a glosa devem ser priorizados, uma vez que o português sinalizado é insuficiente para a compreensão completa por parte dos surdos. Normalmente, o dicionário de sinais é expandido por uma equipe de especialistas em LIBRAS e TI pertencentes ao projeto, porém, sabe-se que a quantidade de sinais a ser confeccionados é muito grande. Desse modo, pode-se perceber o quanto um componente com função de WikiLIBRAS é importante, pois permite que pessoas externas ao projeto possam contribuir com a criação de novos sinais. Por fim, quanto mais opções de formatos de entrada e plataformas de execução o projeto oferecer, mais formatos de entrada e plataforma de execução o serviço DAaaS poderá oferecer. Portanto, a partir das informações colhidas e projetadas na tabela 3.1, foi concluído que para o presente trabalho a melhor opção é o sistema de tradução automática VLIBRAS.

¹T = texto, A = áudio, V = vídeo, L = legenda

²W = *Web*, M = *mobile*

³O código do VLIBRAS foi disponibilizado para o presente trabalho.

3.2 Computação em Nuvem e Processamento Multimídia

O que permite a oferta do DAaaS como um serviço robusto e transparente é a escalabilidade que uma infraestrutura de nuvem pode oferecer. A capacidade de escalabilidade é um dos artifícios que alguns sistemas distribuídos utilizam para tolerar falhas tanto em nível de *software* como em nível de *hardware*, sendo este último por intermédio da alta disponibilidade dos recursos. Uma vez que o serviço ofertado (DAaaS) será escalável e tolerante a falhas, é importante pesquisar alguns trabalhos de referência para posterior comparação das técnicas de escalabilidade e tolerância a falhas utilizadas. Adicionalmente, também é interessante investigar sistemas de processamento multimídia implantados na nuvem, com o objetivo de comparar e aprimorar as características do presente trabalho. Para tal, foram utilizadas as cadeias de busca “*fault tolerance AND cloud computing*”, “*scaling AND cloud computing*”, e “*multimedia cloud computing*” no Google acadêmico e portal de periódicos CAPES, resultando em vários trabalhos dos quais os principais são detalhados a seguir.

3.2.1 Escalabilidade na Nuvem

Adaptive Application Scaling for Improving Fault-Tolerance and Availability in the Cloud (RADHAKRISHNAN, 2012)

Neste trabalho, Radhakrishnan (2012) propõe uma estratégia de escalonamento que aproveita a elasticidade provida pela nuvem para mitigar os efeitos provenientes da exaustão de recursos em uma instância de processamento. Anomalias ocorridas em tempo de execução, falhas na infraestrutura, e erros operacionais decorrentes de uma aplicação podem exaurir os recursos, resultando em gargalos que afetam adversamente todas as outras aplicações/processos compartilhando tais recursos. O autor afirma que, geralmente, a estratégia de monitoramento de recursos (CPU, memória RAM, etc.) por si só não é suficiente, uma vez que amostras de dados de uso dos recursos não provêm garantia consistente de seu desempenho.

A estratégia proposta trata o sistema distribuído como uma rede de servidores que recebem fluxos de requisições com intensidades (o autor chama de fluido) variadas como funções contínuas do tempo, usando técnicas de enfileiramento. O modelo “*Flow Dynamics*”, proposto pelo autor, busca coletar um conjunto de informações em espaços de tempos que são utilizadas para determinar se é preciso escalar, e a quantidade necessária de novas instâncias

para que o sistema atenda as requisições dentro dos parâmetros de QoS delimitados. Todos os servidores dessa rede têm filas individuais, e as métricas para escalonamento são baseadas no “fluido” de requisições nos servidores: capacidade de vazão do fluido, e tempo médio de espera das requisições na fila. De forma geral, as requisições funcionam como fluidos, podendo ter diferentes consistências, e a rede de servidores de processamento funcionam como filtros, onde a capacidade de vazão desse filtro se dá em detrimento da consistência desse fluido. Portanto, Radhakrishnan (2012) propõe um escalonamento baseado na “consistência” desse fluido, que por sua vez varia de acordo com a quantidade de recursos e capacidade de processamento nos servidores no contexto de determinada aplicação.

O algoritmo para escalonamento classifica um servidor em um dos estados mutuamente exclusivos: sub-saturado, saturado, sobre-saturado. No estado sub-saturado, a capacidade do servidor excede os recursos de processamento necessário. Um servidor saturado é aquele cuja capacidade está compatível com a necessidade de recursos para processar o fluido, ou seja, sua fila contém uma pequena quantidade de requisições. O estado sobre-saturado indica que existe uma quantidade significativa de fluido na fila, esperando para ser processado.

Do ponto de vista das despesas operacionais, para a empresa provedora do serviço/aplicação, é preferível operar servidores no modo saturado ou até levemente sobre-saturado, desde que as métricas de QoS sejam atendidas. Portanto, a estratégia de escalonamento proposta é computar para cada servidor a capacidade de vazão do fluido, e o tempo médio de espera nas filas, dentro de um intervalo de tempo, para então, em detrimento desses valores e dos recursos disponíveis, decidir entre as seguintes ações:

1. redirecionar o parte do fluido para outro servidor na nuvem;
2. requisitar recursos adicionais da nuvem para o servidor;
3. criar novos servidores;
4. combinar instâncias de servidores para conservar recursos.

O escalonamento dinâmico proposto protege a aplicação contra as falhas que ocorrem por escassez de recursos, e alta disponibilidade. O algoritmo “*Flow Dynamics*” constitui uma ótima estratégia de escalonamento automático para sistemas implantados na nuvem. O serviço DAaaS possui uma estratégia de escalonamento semelhante, que busca explorar a

capacidade máxima de *hardware* oferecido pelas instâncias de processamento, e baseia suas políticas de escalonamento no tamanho de uma fila única de requisições. Adicionalmente, a fila introduzida no DAaaS permite um tratamento de falhas mais rigoroso no nível de *software*, onde as requisições que falharem podem ser reprocessadas por uma outra instância em pleno funcionamento, e só são removidas da fila quando seu processamento ocorrer sem falhas. Maiores detalhes sobre a estratégia de escalonamento do DAaaS serão apresentados na seção 4.2.2 do Capítulo 4.

3.2.2 Tolerância a Falhas na Nuvem

Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing (BALA; CHANA, 2012)

Para minimizar o impacto de falhas na execução de aplicações e sistemas, é preciso antecipá-las e lidar proativamente com as mesmas. Técnicas de tolerância a falhas são usadas para agir de maneira apropriada depois que elas ocorram. A seguir, discutiremos algumas técnicas de tolerância a falhas em computação em nuvem abordadas no trabalho, e faremos um paralelo com as técnicas utilizadas no DAaaS.

Os principais benefícios de implementar tolerância a falhas em computação em nuvem são: recuperação da falha, atenuação dos custos financeiros e melhora do desempenho do sistema. Bala e Chana (2012) descrevem algumas técnicas de tolerância a falhas classificando-as em duas políticas: tolerância a falhas reativa e proativa. Tais técnicas podem ser usadas tanto em nível de tarefas quanto de controle de fluxo.

Políticas de tolerância a falhas reativas reduzem os efeitos das mesmas em aplicações quando elas efetivamente já ocorreram. As principais técnicas baseadas nessa política são descritas a seguir (BALA; CHANA, 2012):

Checagem de ponto/Reiniciar: quando uma tarefa falha, ela pode ser reiniciada a partir de um ponto previamente checado com sucesso, ao invés de recomeçar completamente. É uma técnica eficiente para aplicações com execuções demoradas;

Replicação: várias réplicas são executadas com diferentes recursos, e a execução só é considerada completa quando as replicações terminarem seus processamentos;

Migração de jobs: se alguma tarefa falhar, ela pode ser migrada para outra instância de processamento;

Tentar novamente: é uma das mais simples que consiste em tentar processar a tarefa que falhou no mesmo recurso da nuvem;

Resubmissão de tarefas: é a técnica mais amplamente utilizada, e consiste em resubmeter a tarefa para o mesmo recurso ou para um diferente;

Tratamento de exceção definida pelo usuário: o usuário especifica um tratamento particular para a falha de uma tarefa específica;

Resgate do controle de fluxo: essa técnica permite que o controle de fluxo continue até que se torne impossível de ir adiante sem recuperar a tarefa que falhou.

Políticas de tolerância a falhas proativas evitam a recuperação de falhas e erros através da prevenção das mesmas, e substituem os componentes que suspeitam ser defeituosos por outros que estejam em pleno funcionamento. As principais técnicas baseadas nessa política são descritas a seguir (BALA; CHANA, 2012):

Rejuvenescimento de software: consiste em programar o sistema para reinícios periódicos, liberando e renovando todos os recursos;

Self-healing: essa estratégia consiste em monitorar o funcionamento dos servidores, e caso algum deles pare de funcionar, o algoritmo cria uma nova instância com a mesma imagem da aplicação. Essa técnica garante a alta disponibilidade do serviço provido;

Migração preemptiva: consiste em um mecanismo de controle que usa um laço trazendo *feedbacks* de tempos em tempos, onde a aplicação é constantemente monitorada e analisada;

O serviço DAaaS usa uma técnica de tolerância a falha reativa (resubmissão de tarefas) e uma proativa (*self-healing*). Na técnica reativa o sistema utiliza a fila de requisições para resubmeter as que falharem. Quando uma instância começa a processar uma requisição da fila, a mesma é marcada como “invisível” até que essa instância termine seu processamento e explicitamente a remova da fila. Caso contrário, depois de um tempo de invisibilidade

padrão⁴, a requisição será marcada automaticamente como visível, e pode ser reprocessada pela mesma instância ou por uma diferente, já que as instâncias “competem” para processar essa fila única de requisições.

Maiores detalhes sobre as estratégias de tolerância a falhas implementadas no DAaaS serão apresentadas na seção 4.2.1 do Capítulo 4.

3.2.3 Sistemas de Processamento Multimídia Implantados na Nuvem

Multimedia Cloud Computing (ZHU et al., 2011)

Este trabalho introduz os principais conceitos de computação em nuvem para processamento multimídia. Especificamente, é proposto um *framework* para processamento multimídia que aproveita a computação em nuvem para fornecer aplicações e serviços multimídia através da Internet com provisionamento de QoS. O trabalho explica como é possível prover suporte a QoS, processamento paralelo distribuído, armazenamento, e balanceamento de carga, para que as várias aplicações e serviços multimídia os utilizem de maneira otimizada.

Para demonstrar algumas das técnicas propostas no *framework*, o trabalho utiliza o *Photosynth* para exemplificar como o processamento multimídia paralelo funciona e supera a abordagem tradicional. *Photosynth* (<http://www.photosynth.net/>) é um *software* que analisa fotografias digitais e gera modelos tridimensionais. Zhu et al. (2011) explicam que os usuários são capazes de gerar seus próprios modelos usando o *software* cliente e depois realizar o *upload* da foto para o sítio do *Photosynth*. Portanto, o processamento das imagens é feita em um computador local.

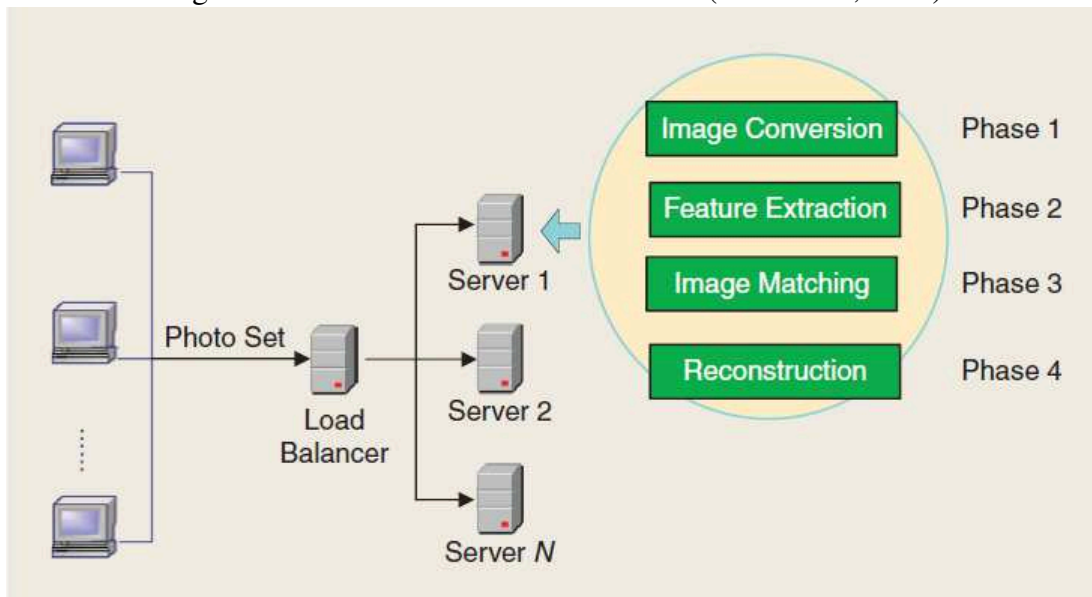
As tarefas executadas pelo programa *Photosynth* são conversão de imagens, extração de características, combinação de imagens, e reconstrução. Na abordagem tradicional, as quatro tarefas supracitadas são executadas sequencialmente em um cliente local. A proposta é que essas quatro tarefas sejam executadas na nuvem. Tal modelo seria extremamente importante para dispositivos móveis devido as restrições de bateria e baixas capacidades de processamento.

Para reduzir o tempo de processamento quando o número de usuários for grande, Zhu et al. (2011) propõem uma arquitetura paralela baseada em nuvem. Tal arquitetura possui um

⁴tempo previamente calculado que supõe-se que qualquer requisição já deveria ter sido atendida

balanceador de carga que permitirá que o algoritmo *Photosynth* seja executado em *pipelines* paralelos. A paralelização pode ser executada a nível de usuários ou de tarefas. Na Figura 3.1, é ilustrado o paralelismo a nível de usuário no qual todas as tarefas do mesmo são alocadas para processamento em um servidor, porém as tarefas de outros usuários podem ser executadas simultaneamente de forma paralela.

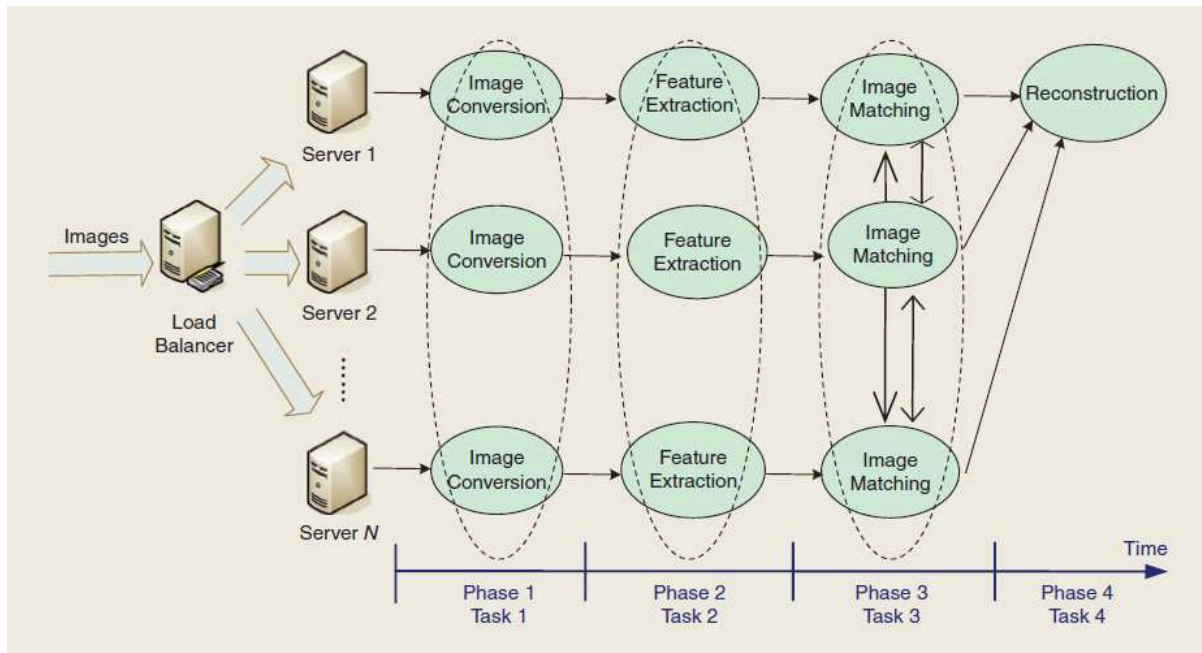
Figura 3.1: Paralelismo a nível de usuário (ZHU et al., 2011)



A Figura 3.2, é ilustrado o paralelismo a nível de tarefa na qual todas as tarefas de um usuário são alocadas em N servidores para processamento paralelo. Mais especificamente, as tarefas de conversão de imagem, extração de características, e combinação de imagens são processadas em servidores diferentes. De acordo com a Figura 3.2, quando o atraso proveniente da transmissão é omitido, a conversão da imagem, extração de características, e combinação de imagens podem economizar N vezes menos tempo de processamento usando a paralelização a nível de tarefa com N servidores, do que o caso tradicional com apenas um servidor.

Zhu et al. (2011) realizaram dois experimentos com o *software Photosynth* utilizando o paralelismo a nível de tarefas. Foram submetidas 200 e 400 imagens um *cluster* com dois nós de processamento, e posteriormente em outro *cluster* com nove nós de processamento. Para o caso onde havia 2 nós de processamento houve *speedup* de 1,65 vezes o processamento sequencial tradicional, e para o caso onde havia nove servidores houve *speedup* de 4,25

Figura 3.2: Paralelismo a nível de tarefa (ZHU et al., 2011)



vezes o processamento sequencial tradicional.

No Capítulo 4, é possível observar que a arquitetura proposta se assemelha com a arquitetura detalhada por Zhu et al. (2011) que realiza o processamento paralelo a nível de usuário. O paralelismo a nível de usuários foi escolhido pois é suficiente para prover escalabilidade e tolerância a falhas, sem a necessidade de grande intrusão no sistema VLIBRAS. Essa proposta pode reduzir o tempo de processamento, uma vez que a inserção do balanceador de carga e novos nós de computação elevam a capacidade de processamento do sistema. Adicionalmente, o presente trabalho se utiliza da capacidade de provisionamento dinâmico, alocando e desalocando recursos de acordo com a demanda, fazendo que o uso eficiente do *hardware* proporcione economia financeira.

MediaCloud: A New Paradigm of Multimedia Computing (HUI; YANG, 2012)

Neste trabalho, Hui e Yang (2012) propõem o *MediaCloud* como um novo paradigma de computação multimídia que integra o conceito de computação em nuvem para lidar com aplicações e serviços multimídia de forma eficaz e eficiente. Para prover suporte a aplicações com enfoque em conteúdos multimídia, o *MediaCloud* tem como principais desafios a heterogeneidade, escalabilidade e provisionamento de QoS para serviços multimídia:

Heterogeneidade: abrange desde a heterogeneidade de dispositivos à heterogeneidade de rede. Dispositivos móveis são exemplos claros de heterogeneidade, pois possuem diversos tipos de rede de acesso (3G, 4G, *wireless*), e diversas capacidades de *hardware* de dispositivos (tamanho de tela, capacidade de bateria, capacidade de processamento) que as aplicações multimídias devem levar em conta;

Escalabilidade: geralmente, serviços multimídia baseados em nuvem têm que gerenciar grandes números de usuário acessando aplicações e conteúdos concorrentemente, e lidar com grandes picos e quedas de demandas sem realizar gastos desnecessários;

Provisionamento de QoS para serviços multimídia: aplicações e serviços multimídia geralmente trabalham com grandes conjuntos de dados que possuem requisitos de QoS específicos e rigorosos. Latência, por exemplo, é um requisito muito importante para serviços de vídeo-conferência em tempo real. As nuvens atuais não são projetadas para aplicações multimídia com requisitos rigorosos. Portanto, oferecer serviços multimídia com provisionamento de QoS não é tão simples.

Os três desafios citados por Hui e Yang (2012) também são enfrentados no presente trabalho. Uma vez que o serviço/API esteja pronto para uso por parte de terceiros, o mesmo poderá ser utilizado de forma completamente heterogênea por dispositivos móveis ou computadores convencionais; o serviço é oferecido de forma pública e irrestrita, tornando obrigatório o tratamento da escalabilidade na nuvem.

Para solucionar tais desafios Hui e Yang (2012) apresentam uma arquitetura sobre computação em nuvem que provê serviços multimídia escaláveis: *MediaCloud*. A proposta de Hui e Yang (2012) é abstrair os serviços de computação em nuvem para serviços multimídia. Para tanto, o autor utiliza técnicas difundidas de computação em nuvem como virtualização, para prover serviços de qualidade focados nos requisitos de aplicações multimídia.

O trabalho exhibe a implementação de uma aplicação multimídia sobre a arquitetura do *MediaCloud*, e compara com outros programas similares que não são implantados na arquitetura do *MediaCloud*. A arquitetura do *MediaCloud* apresenta ganhos tanto em termos de eficiência como escalabilidade, inclusive para uma versão cliente executada em *smartphones*.

O presente trabalho não é implementado sobre uma arquitetura de nuvem modificada para atender os requisitos específicos de aplicações multimídias. Contudo, ao implantar o serviço

DAaaS em uma infraestrutura de nuvem, todos os desafios citados e tratados por Hui e Yang (2012) são enfrentados, uma vez que o DAaaS também lida com conteúdo multimídia. Em contrapartida, o fato da implantação do sistema DAaaS ser diretamente na nuvem dá mais liberdade na forma de solucionar os problemas, através do gerenciamento das camadas de baixo nível da nuvem.

3.3 Considerações

Nesse Capítulo foram apresentados os principais trabalhos relacionados presentes na literatura, categorizados em dois sub-grupos:

1. trabalhos relacionados à disponibilização de acessibilidade para surdos brasileiros nas TICs;
2. trabalhos que abordem técnicas de computação em nuvem para oferecer escalabilidade e tolerância a falhas, e sistemas de processamento multimídia implantados na nuvem.

Ao pesquisar o primeiro grupo, resumizamos as principais ferramentas de tradução língua portuguesa -> LIBRAS com o objetivo de embasar a escolha do tradutor que será utilizado como núcleo do DAaaS. A partir das informações colhidas e projetadas na tabela 3.1, foi concluído que para o presente trabalho a melhor opção é o sistema de tradução automática VLIBRAS, devido aos vários formatos de entrada disponível, plataformas de execução, e forma da tradução utilizando a glosa. O segundo grupo de trabalhos pesquisados serviu para relatar as principais técnicas utilizadas para escalonamento de recursos e tolerância a falhas na nuvem, e compará-las com as técnicas utilizadas no DAaaS. Mais detalhes dessas técnicas serão explicitadas no próximo Capítulo. Por último, descrevemos detalhes da implementação e vantagens de dois sistemas de processamento multimídia implantados na nuvem, relacionando-os ao sistema DAaaS.

No próximo Capítulo será apresentada a arquitetura proposta para fornecer o sistema DAaaS de maneira escalável e tolerante a falhas, por meio da computação em nuvem.

Capítulo 4

Arquitetura Proposta

Sistemas centralizados apresentam várias desvantagens quando comparados a sistemas distribuídos. A solução atual para o sistema de tradução automática VLIBRAS é baseada em um sistema centralizado, e, portanto, pode ser otimizada em vários aspectos: tempo de resposta para uma requisição, capacidade de vazão (diretamente relacionada ao tempo de resposta), e capacidade de tolerância a falhas. Para tanto, é necessária uma mudança na arquitetura do VLIBRAS, tornando seu sistema centralizado um sistema distribuído.

As características do modelo de computação em nuvem, especialmente o provisionamento dinâmico de recursos, tornam esta uma excelente alternativa para a implantação do VLIBRAS de forma distribuída. Este modelo vem sendo amplamente adotada nas corporações e até pelos governos devido a sua facilidade e flexibilidade de uso (ZHANG; CHENG; BOUTABA, 2010). Adicionalmente, fatores como escalabilidade automática de acordo com a demanda e pagamento dos serviços por tempo de uso convergem para maior economia por parte da organização.

Uma das intenções do presente trabalho é, implantar o VLIBRAS em uma arquitetura dinâmica e tolerante a falhas. A capacidade de provisionamento dinâmico permitirá o sistema alocar recursos para atender grandes cargas de requisição e readaptar-se quando a demanda diminuir, economizando recursos. Essa capacidade de escalonamento automático é uma das características que atenua a probabilidade de ocorrência de falhas, através da alta disponibilidade dos recursos. Além disso, o trabalho utiliza uma técnica de tolerância a falhas reativa (resubmissão de tarefas) e uma proativa (*self-healing*), que serão descritas ao longo do Capítulo.

Nas seções a seguir, iremos detalhar a antiga arquitetura do VLIBRAS e a atual arquitetura proposta (DAaaS), com objetivo de comparar as vantagens e desvantagens que cada uma apresenta. É preciso ressaltar que para fins de disponibilização do serviço, foi preciso implementar uma interface RESTful para o VLIBRAS, que permitirá que terceiros utilizem o DAaaS. Na última seção detalharemos a incorporação desta API ao VLIBRAS, além de suas especificações para uso.

4.1 VLIBRAS - Arquitetura Centralizada

Não são todas as organizações de TI que utilizam o paradigma de computação em nuvem como infraestrutura base para seus serviços. Na verdade, muitas empresas por possuírem parte dessa infraestrutura de *hardware* a longo prazo, as vezes criam resistência para realizar essa transição na forma de oferecer seus serviços.

A antiga arquitetura do VLIBRAS era constituída por um sistema centralizado, em um único servidor convencional. Este é o modo mais comum de implantação, praticado pelas empresas de TI que não adotam o paradigma de computação em nuvem. Tal arquitetura apresenta algumas desvantagens quando comparada às que são implantadas na nuvem:

1. Incapacidade de alocar novos recursos para atender altos picos de demanda;
2. Incapacidade de desalocar recursos para economizar quando o serviço estiver com baixa demanda;
3. Incapacidade de tolerar determinados tipos de falhas:
 - (a) se o servidor de produção por algum motivo parar de funcionar (problema de rede, falta de energia, ou problemas de *hardware*) o serviço fica indisponível;
 - (b) se a demanda estiver muito grande, ou seja, maior que a capacidade de processamento do servidor, há a possibilidade de falhas no atendimento de algumas requisições, deixando alguns clientes do serviço sem resposta;
4. Não oferece outras vantagens que o paradigma de computação em nuvem provê:
 - (a) não há necessidade de espaço físico, energia, refrigeração, e manutenção da infraestrutura de *hardware*;

- (b) fácil implantação e configuração do serviço, em diferentes lugares do mundo;
- (c) pagamento exclusivamente perante uso, não havendo necessidade de manter máquinas ociosas.

Na Figura 4.1, é ilustrado o funcionamento desta arquitetura. O VLIBRAS é um programa *Web*, mas, para fins de comparação, vamos supor que ele seja fornecido através de uma API para desenvolvedores. Portanto, os usuários finais só utilizariam este serviço através de algum serviço intermediário, como por exemplo, um sítio da Internet que criasse um *plugin* que utilizasse o VLIBRAS. A seguir, é detalhado todo o processo que o usuário final deveria esperar para ter seu conteúdo traduzido para LIBRAS através de, por exemplo, um serviço de VoD:

1. O usuário escolhe um vídeo do sistema de VoD para assistir, e requisita a opção de acessibilidade;
2. O sistema de VoD envia uma requisição e as respectivas entradas (vídeo/legenda) a ser traduzida pelo VLIBRAS. Tal requisição deve respeitar uma interface definida pelo serviço, e pode ser enviada no formato JSON ou XML;
3. O servidor executa a tradução automática e envia a tradução de volta ao sistema de VoD;
4. O sistema de VoD exibe o vídeo com sua respectiva tradução de maneira sincronizada.

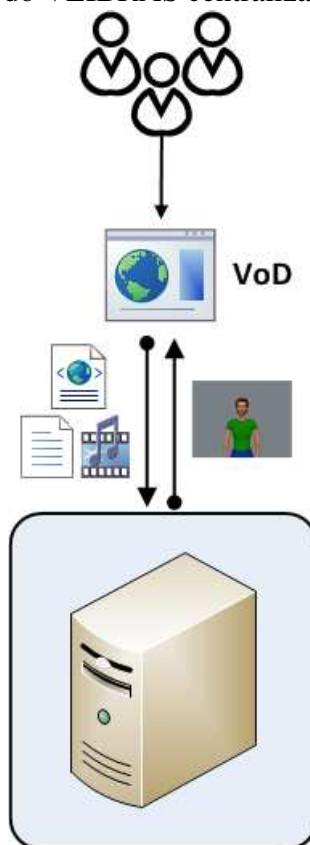
4.2 DAaaS - Arquitetura Distribuída

A arquitetura proposta para o DAaaS foi projetada para ser implantada em uma infraestrutura de computação em nuvem. O objetivo desta arquitetura é incorporar as seguintes funcionalidades:

Descentralização do fluxo de rede: a existência de vários nós de processamento implica em vários fluxos de rede entre os clientes e as instâncias de processamento;

Provisionamento dinâmico de recursos: Quando em momentos de sobrecarga, a escalabilidade sob demanda permite que o DAaaS tenha recursos disponíveis para processar as requisições através da inclusão de novas instâncias de processamento;

Figura 4.1: Arquitetura do VLIBRAS centralizada em um único servidor

**Tolerância a falhas:**

Alta disponibilidade dos recursos: os recursos são sempre disponibilizados com redundância e em diferentes zonas de disponibilidade;

Em nível de *software*: tolerância a falhas reativa por resubmissão de tarefas, e proativa por meio de *self-healing*;

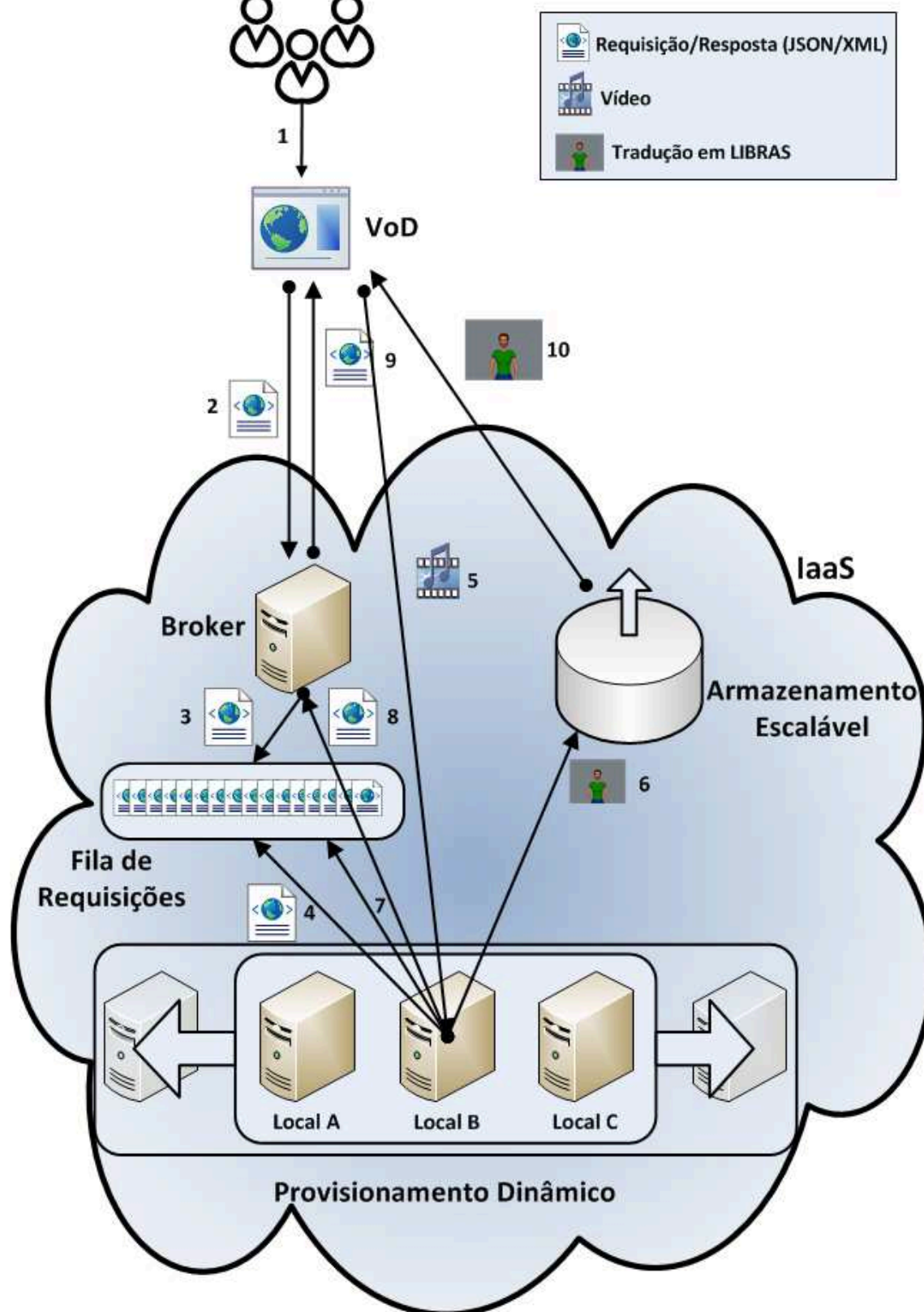
Observando a Figura 4.1, é possível constatar que existe um único canal de fluxo de rede para que os vídeos de entrada (ou legenda ou texto) e de tradução trafeguem. Tal fato converge para elevar a probabilidade de haver um gargalo de rede, aumentando o tempo de resposta para as requisições. A arquitetura proposta (Figura 4.2) busca diminuir o gargalo de rede uma vez que existirá um canal de rede para cada instância de processamento, e, portanto, vários canais de tráfego de vídeos para o sistema como um todo. Ao invés do sistema de VoD enviar primariamente o arquivo de vídeo para tradução (como acontece na arquitetura centralizada), envia apenas uma requisição (arquivo XML/JSON) que contém o

endereço daquele vídeo, e então a instância de processamento estabelece uma conexão direta para realizar o *download* do vídeo. Esta característica do sistema evita a sobrecarga no canal de rede em que se insere o *Broker*, uma vez que os arquivos XML/JSON possuem carga extremamente inferior a arquivos de vídeo.

Na Figura 4.2, é ilustrado o funcionamento da arquitetura proposta. Ela é basicamente composta por quatro componentes: *Broker*, Fila de Requisições, instâncias de processamento VLIBRAS e armazenamento escalável. O *Broker* é o *endpoint* de entrada do DAaaS, sendo responsável primariamente pelo gerenciamento das requisições. A Fila de Requisições mantém um registro das requisições a serem processadas e em processamento. As instâncias de processamento VLIBRAS são VMs com o sistema VLIBRAS instalado, prontas para receber e processar requisições. O armazenamento escalável é responsável por guardar o resultado das traduções, de onde os usuários finais poderão recuperar. Para explicar o funcionamento desta arquitetura será utilizado o exemplo em que o sistema de VoD é o cliente do DAaaS. O processo completo para que um sistema de VoD requisição a tradução de algum conteúdo obedece a sequência de passos ordenada e identificada pelos números na Figura 4.2.

1. O usuário escolhe um vídeo do sistema de VoD para assistir e requisita a opção de acessibilidade;
2. O sistema de VoD envia uma requisição (arquivo JSON ou XML em conformidade com a API do DAaaS) de tradução ao *Broker* do sistema DAaaS;
3. O *Broker* adiciona essa requisição a uma fila de requisições, e notifica as instâncias de processamento para que elas saibam que existe uma nova requisição na fila;
4. As instâncias que possuem recursos livres, irão competir para processar as requisições na fila. Cada requisição é alocada de forma atômica por uma única instância, nesse caso, a instância B;
5. A instância B realiza o *download* do vídeo e dá início à tradução automática;
6. Uma vez que a tradução para LIBRAS seja concluída, o vídeo de tradução é transferido para o serviço de armazenamento de arquivos escalável;
7. A instância envia um sinal para a Fila de Requisições remover aquela requisição específica, uma vez que tenha sido processada com sucesso;

Figura 4.2: DAaaS - Arquitetura Distribuída



8. Uma mensagem contendo o endereço do vídeo de tradução em LIBRAS é enviado para o *Broker*;
9. Essa mensagem é enviada para o sistema de VoD;
10. O sistema de VoD realiza o *download* do vídeo de tradução em LIBRAS e exibe para o usuário final de maneira sincronizada com o vídeo original.

4.2.1 Tolerância a Falhas

Alta Disponibilidade dos Recursos

A existência de redundância de recursos em localizações geográficas diferentes provê tolerância a falhas por meio de alta disponibilidade. De acordo com a Figura 4.2, as instâncias de processamento estão situadas em diferentes locais físicos. É possível configurar o provisionamento dinâmico para criar novas instâncias em diferentes localizações geográficas, visando equilibrar a quantidade de instância nestes locais (e.g., 2 instâncias no “local A”, 2 instâncias no “local B”, e 2 instância no “local C”). Se porventura alguma unidade de processamento falhar, o DAaaS não ficará indisponível, uma vez que outros nós de processamento podem assumir a carga do nó defeituoso até que o problema seja resolvido. Deste modo, evita-se que problemas com rede ou energia em determinada localização, ou até mesmo falhas de *hardware*, interfiram na disponibilidade do DAaaS.

Em Nível de *Software*

Na antiga arquitetura centralizada, as falhas eram irreversíveis, ou seja, se ocorressem não eram passíveis de recuperação. A tolerância a falhas em nível de *software* permite a recuperação de uma falha ocorrida, além de sua prevenção por antecipação.

Os componentes responsáveis pela tolerância a falha em nível de *software* são o *Broker* e a Fila de Requisições. Com relação a tolerância a falha, as principais atividades do *Broker* são:

Gerenciamento de requisições: toda requisição chega primeiro ao *Broker*, que é encarregado de enviá-la para a Fila de Requisições;

Notificação: assim que o *Broker* adiciona uma nova requisição à Fila de Requisições, seu próximo passo é notificar todos os nós de processamento para que eles saibam o momento correto de consumir a Fila de Requisições;

Provisionamento dinâmico de recursos: baseado na quantidade de requisições na Fila de Requisições, o *Broker* adiciona ou remove novos nós de processamento. O *Broker* também faz checagens periódicas para garantir que todos os nós de processamento estejam em pleno funcionamento, removendo os defeituosos e adicionando novos quando necessário.

O serviço DAaaS usa uma técnica de tolerância a falha reativa por resubmissão de tarefas. A técnica reativa proposta consiste em resubmeter as requisições que falharem à Fila de Requisições. Para tal, todas as requisições desta Fila de Requisições são marcadas com uma *tag* “visível” ou “invisível”. Todas as requisições inseridas na Fila de Requisições são marcadas por padrão como visível, para que todas as instâncias possam acessá-las. Quando o *Broker* adiciona novas requisições a essa Fila de Requisições, sua próxima tarefa é notificar as instâncias de processamento para avisar que a fila possui novos itens a serem consumidos. Assim que uma instância começa a processar uma requisição da Fila de Requisições, a mesma é marcada como “invisível” até que essa instância termine seu processamento e explicitamente a remova da fila.

Testes preliminares (Capítulo 5) foram realizados para definir quantas requisições simultâneas cada instância consegue processar com sucesso, e qual o pior tempo para processamento em situações de estresse. Consideramos como falha o caso em que uma instância não conseguir processar a requisição dentro desse pior tempo estipulado. Este pior tempo é utilizado como tempo de invisibilidade padrão, ou seja, se o processamento de alguma requisição for maior do que o pior tempo pré-calculado a requisição será automaticamente marcada como visível e será reprocessada por outra instância ou até pela mesma. Portanto, há a possibilidade de acontecer a “falha real”, onde o processamento da requisição realmente falhou, mas também há a possibilidade da “falha falsa”, quando uma instância demorar mais tempo para processar uma requisição do que o pior tempo. Quando ocorrer a “falha falsa” a requisição será processada duas ou mais vezes, onde o primeiro processamento pode até ser concluído com sucesso, mas por demorar muito é desconsiderado pelo DAaaS, e o último

processamento terminará antes do pior tempo.

Quando uma requisição falhar e tornar-se visível novamente, não haverá mais nenhuma notificação exclusiva do *Broker* às instâncias avisando que aquela requisição específica está mais uma vez visível. Contudo, as instâncias podem consumir a requisição que falhou através de outras notificações do *Broker* para outras requisições, ou através de um laço de checagem de requisições (15 segundos) que as instâncias de processamento implementam para consumir novas requisições independente da notificação do *Broker*. O modelo de processo de negócios para tolerância a falhas reativa por meio de resubmissão de tarefas é detalho na Figura 4.3, através de um diagrama BPMN (*Business Process Modeling Notation*).

A técnica proativa também é implementada no *Broker*, através do provisionador dinâmico de recursos, que é encarregado de alocar ou desalocar recursos diante das métricas especificadas. Uma das funções deste provisionador é verificar se alguma das instâncias não está mais em funcionamento, através de requisições HTTP às máquinas supostamente ativas, e as repor caso elas aparentem estar defeituosas.

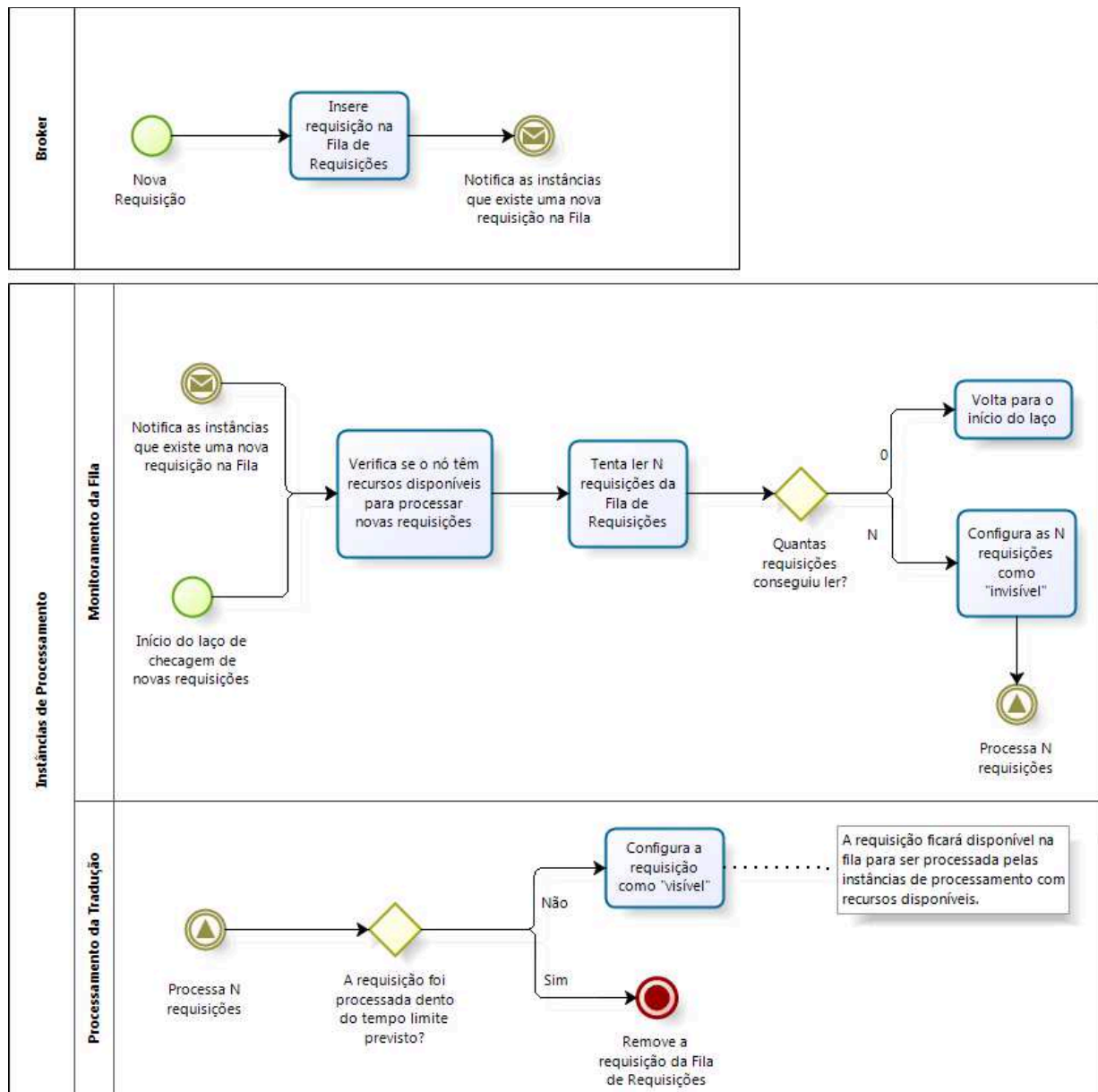
4.2.2 Provisionamento Dinâmico de Recursos

A estratégia utilizada para provisionamento dinâmico de recursos baseia-se na Fila de Requisições. Deste modo, testes preliminares foram realizados para definir quantas requisições simultâneas cada tipo de instância utilizada consegue processar, e qual o pior tempo para processamento em situações de estresse. Baseado na quantidade máxima de requisições simultâneas, no tamanho da Fila de Requisições, e na quantidade de instâncias de processamento ativas, é possível calcular se o DAaaS está sobrecarregado ou subutilizado, para criar novas instâncias ou remover algumas.

A tendência é que as instâncias de processamento estejam sempre sobrecarregadas (caso haja requisições na Fila), uma vez que o *Broker* sempre notifica todas as instâncias quando novas requisições são inseridas na fila, além da existência de um laço de checagem (15 segundos) de novas requisições nas instâncias de processamento, que independe do *Broker*. Portanto, as instâncias de processamento estarão sempre “absorvendo” a Fila de Requisições se tiverem recursos disponíveis, ou seja, sempre que não estiverem processando a quantidade máxima de requisições delimitadas.

O diagrama ilustrado na Figura 4.4, detalha os passos do algoritmo de provisionamento

Figura 4.3: Modelo de Processos de Negócios para Tolerância a Falhas Reativa



dinâmico para criar ou terminar as instâncias de processamento VLIBRAS. Seguindo esse modelo de processo de negócios, é possível simular o que aconteceria se, por exemplo, a Fila de Requisições possuísse 1435 requisições, a instância pudesse processar no máximo 15 requisições, e o pior tempo para processamento em situações de estresse fosse 30 minutos. Lembrando que o DAaaS inicia com 2 instâncias de processamento, que é a quantidade mínima, o algoritmo de escalonamento executaria os seguintes passos:

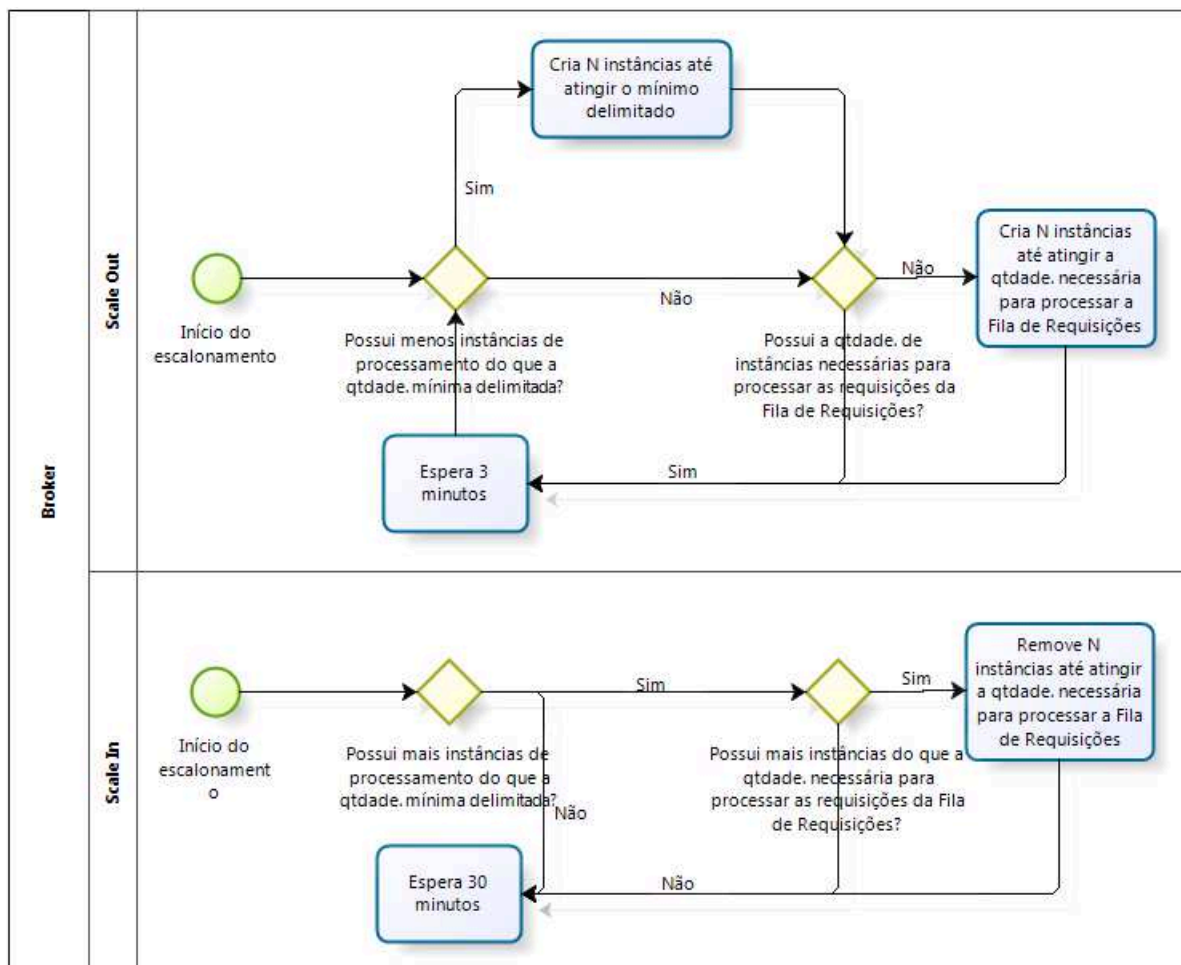
1. "O DAaaS possui menos instâncias de processamento do que a quantidade mínima

- delimitada?” **Não**, o DAaaS tem 2 instâncias disponíveis.
2. “O DAaaS possui a quantidade de instâncias necessárias para processar a Fila de Requisições?” **Não**, o DAaaS possui apenas 2 instâncias, que provavelmente não terem capacidade para atender todas as requisições em tempo hábil.
 3. “O DAaaS cria N instâncias até atingir a quantidade necessária para processar a Fila de Requisições.” Abaixo vamos calcular N a partir do código do DAaaS:
 - (a) `int instancesRequired = (int) Math.ceil(requestsOnQueue/maxRequestsPerInstance);`
 - (b) `int instancesRequired = (int) Math.ceil(1435/15);`
 - (c) `int instancesRequired = (int) Math.ceil(95.66);`
 - (d) `int instancesRequired = 96;`
 4. Sabendo que uma instância leva de 90 a 120 segundos para iniciar, 3 minutos é um tempo com folga para que a instância inicie e absorva a quantidade máxima de requisições, nesse caso, 15.
 5. “O DAaaS possui menos instâncias de processamento do que a quantidade mínima delimitada?” **Não**, o DAaaS tem 98 instâncias disponíveis.
 6. Quando o algoritmo de escalonamento chegar neste ponto, “O DAaaS possui a quantidade de instâncias necessárias para processar a Fila de Requisições?”, a resposta será **SIM**, já que as 96 instâncias recém criadas absorveram toda a Fila de Requisições. A menos que nesse meio tempo novas requisições tenham chegado na fila, e que as instâncias ativas não as tenham conseguido processar.

Para redução da quantidade de instâncias, o chamado *scaling in*, o DAaaS esperaria o pior tempo para processamento em situações de estresse. No caso do exemplo anterior, o DAaaS esperaria 30 minutos para calcular quantas instâncias deveriam ser eliminadas a partir do número de requisições na Fila, e da quantidade de instâncias ativas. Quando o *Broker* notifica uma instância para ser removida ela não é imediatamente excluída, porém, a partir desta notificação ela se prepara pra encerrar, ignorando as novas requisições na Fila de Requisições notificadas pelo *Broker* ou pelo laço de checagem. Uma *thread* é ativada para

checar continuamente se as requisições em andamento já finalizaram, e em caso positivo, a *thread* encerra a instância atual. Um detalhe que pode ser melhorado nesse algoritmo é o monitoramento exato de quantas requisições cada instância ativa possui, para ao notificar a remoção de algumas instâncias, fazê-lo de forma crescente, notificando primeiramente as instâncias com menor número de requisições.

Figura 4.4: Modelo de Processos de Negócios para Provisionamento Dinâmico de Recursos



4.3 API DAaaS

A antiga arquitetura do VLIBRAS o concebia como um programa *Web*. Para fins de disponibilização do serviço, foi preciso implementar uma interface RESTful para o VLIBRAS, que permitirá que terceiros utilizem o DAaaS. Essa interface de acesso é a camada superior do

Broker, implementada com a linguagem de programação *Java* e a biblioteca *Jersey*.

A implementação da API DAaaS dá suporte a todos os formatos de entrada que o VLIBRAS aceita: texto, legenda, áudio e vídeo (*closed caption* e áudio). Contudo, a tradução a partir do áudio ou do vídeo ainda está em fase experimental, e não será disponibilizada nesta versão do DAaaS. A tabela 4.1 lista os tipos de traduções disponíveis no VLIBRAS, e seus *status* na API DAaaS. Os códigos fonte presentes no Apêndice A são referentes ao arquivo JSON para tradução a partir de texto e legenda respectivamente.

Tabela 4.1: Tipos de Traduções disponíveis no VLIBRAS, e seus *status* na API DAaaS

Tipo de tradução	Status
texto	disponível
legenda	disponível
áudio	indisponível
áudio do vídeo	indisponível
<i>closed caption</i>	indisponível
legenda + mixagem	indisponível
áudio do vídeo + mixagem	indisponível
<i>closed caption</i> + mixagem	indisponível

Pontos importantes como a autenticação do cliente na API ainda não foram implementados. Para requisitar a tradução ao DAaaS, o cliente só precisa realizar uma requisição HTTP POST para a *url* (184.169.148.166/Broker/requests), passando um arquivo JSON como parâmetro, em um dos formatos previamente citados, e terá como resposta a *url* do vídeo de tradução. Exemplo de uma requisição HTTP POST ao DAaaS pode ser observada abaixo (4.1).

Código Fonte 4.1: Exemplo de uma requisição HTTP POST ao DAaaS

```
1 curl -X POST -d @from-text.json http://184.169.148.166/Broker
   /requests --header "Content-Type:application/json"
```

4.4 Cenários de Uso da API

A API está sendo disponibilizada na AWS, recebendo requisições na *url* `http://184.169.148.166/Broker/requests`. O DAaaS está implantado utilizando instâncias do tipo *m1.small* e é financiado por meio de créditos da AWS para pesquisas científicas. Atualmente, a API possui dois usuários: 1 - sítio do SENAPES; 2 - aplicativo VLIBRAS *mobile*.

O SENAPES é um evento que será aberto ao público e tem como objetivo principal a apresentação e discussão de experiências no desenvolvimento e implantação de tecnologias assistivas para promoção da acessibilidade para pessoas surdas nas TICs. Portanto, haverá grande acesso de pessoas surdas ao sítio e o mesmo será acessível a elas por meio do DAaaS.

Para obter a tradução de um texto no sítio do SENAPES o usuário deve selecionar o texto, e clicar no botão “Traduzir”. Então uma janela de carregamento aparecerá, e quando a tradução estiver disponível será prontamente exibida ao usuário. As Figuras 4.5, 4.6 e 4.7 ilustram esse processo. O *plugin* de tradução para o sítio SENAPES foi desenvolvido pelo LAVID utilizando a API DAaaS.

Figura 4.5: Selecionando o texto



Figura 4.6: Clicando no botão “Traduzir”



O aplicativo VLIBRAS *mobile*, desenvolvido por Luciano Medeiros de Carvalho Júnior (aluno do PPGI-UFPB), é disponibilizado para plataforma *android* e tem dois objetivos: 1 - disponibilizar um meio móvel eficiente de tornar conteúdos acessíveis para surdos; 2 - validar a API DAaaS. Adicionalmente, o aplicativo VLIBRAS *mobile* comprova que o DAaaS pode ser utilizado de forma heterogênea em diferentes meios de disponibilização. O aplicativo permite tradução a partir de texto, legenda, e também a partir do áudio, aplicando um passo adicional de conversão para texto. As Figuras 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, e

Figura 4.7: Avatar traduzindo o texto selecionado



4.14, ilustram o processo de tradução do VLIBRAS *mobile* a partir do texto. O aplicativo está disponível para *download* em <https://s3.amazonaws.com/VLIBRAS-WORKLOAD/vlibras-mobile.apk>. É necessário que o usuário instale um *player* de vídeo para formatos TS, onde o mais recomendado é o VLC.

4.5 Considerações

Nesse Capítulo foram apresentadas a arquitetura centralizada em que o VLIBRAS era implantado, e a arquitetura distribuída proposta pelo presente trabalho. Dentro desse contexto, detalhamos as principais vantagens que a arquitetura distribuída implantada na nuvem apresenta em relação a centralizada: a capacidade de tolerância a falhas e de provisionamento dinâmico de recursos. Ainda foram explicadas as técnicas utilizadas para tolerar falhas, que envolvem alta disponibilidade dos recursos, além de tratamento em nível de *software*, utilizando o *Broker* e a Fila de Requisições. Também explicamos a estratégia utilizada para provisionamento dinâmico de recursos que também tem como principais componentes o *Broker* e a Fila de Requisições, através de uma simulação hipotética de escalonamento aliada a um diagrama BPMN. Por fim, detalhamos o *status* atual da API DAaaS, exemplificando como requisitar uma tradução ao serviço.

Figura 4.8: Tela principal

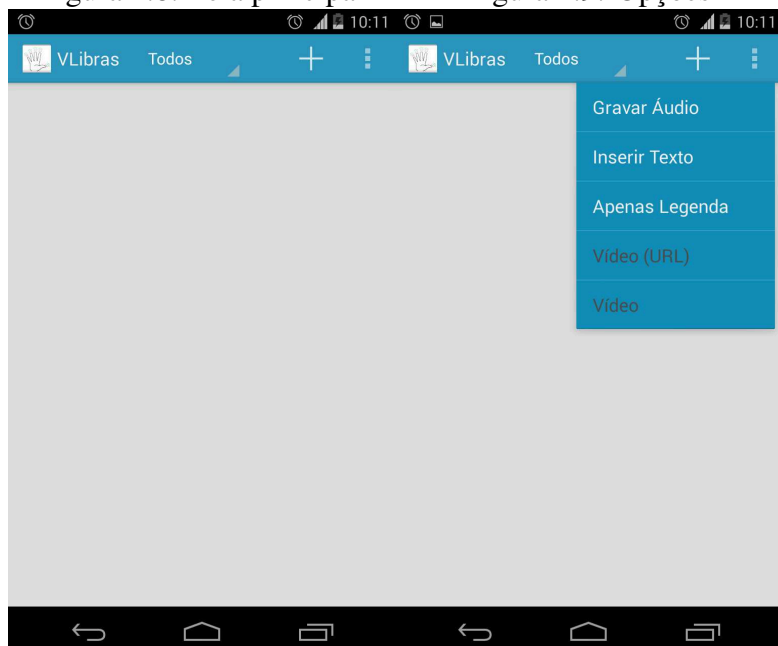


Figura 4.9: Opções

Figura 4.10: Opção: texto

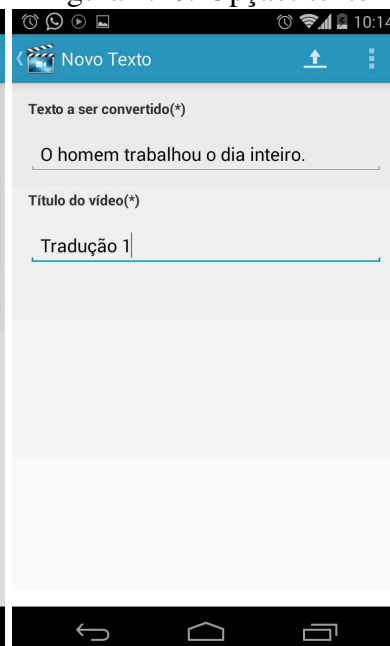


Figura 4.11: Processando

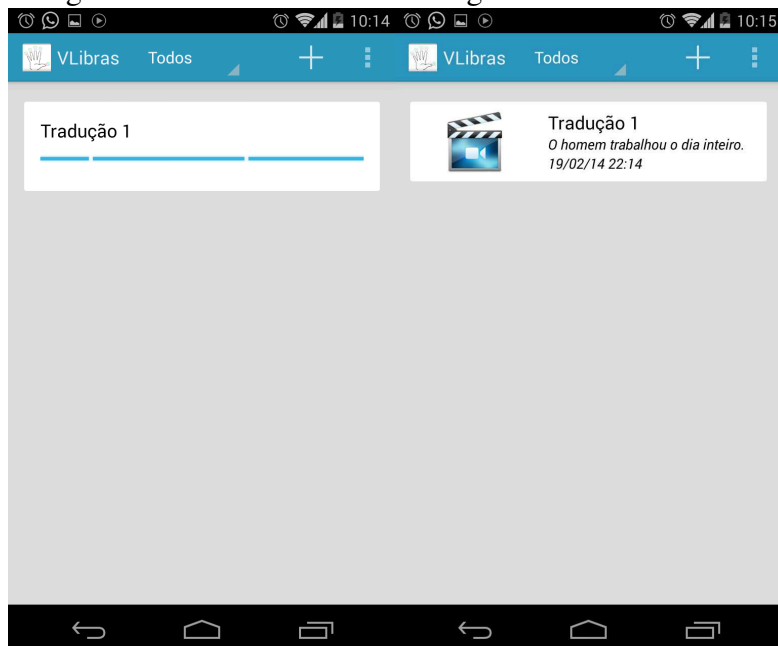


Figura 4.12: Concluída

Figura 4.13: Tocar tradução

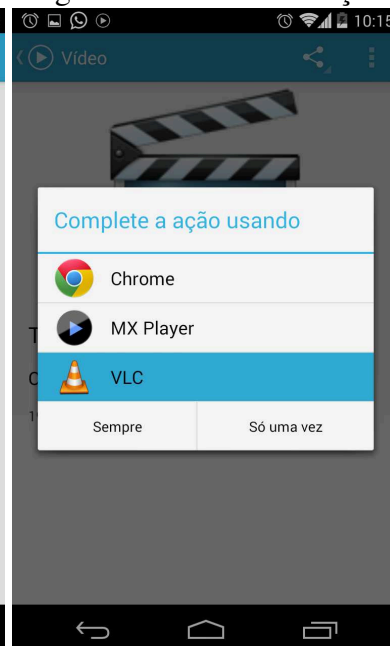


Figura 4.14: Avatar traduzindo o texto selecionado



Para validar a arquitetura proposta e verificar o real funcionamento do serviço é preciso realizar uma análise minuciosa de seu comportamento perante um conjunto de experimentos. O projeto de experimentos tem como principais objetivos avaliar a capacidade de provisionamento dinâmico de recursos e de tolerância a falhas, e será apresentado no próximo Capítulo (5).

Capítulo 5

Experimentos

Nesse Capítulo será detalhado o projeto de experimentos que tem como objetivo validar a arquitetura proposta através da avaliação da capacidade de provisionamento dinâmico de recursos e de tolerância a falhas. Primeiramente, será definida a carga de trabalho a ser utilizada, e serão realizados alguns testes preliminares para descobrir a capacidade máxima de requisições simultâneas que os dois tipos de instância Ec2 utilizadas conseguem atender sem apresentar falhas. Adicionalmente, será escolhido um ponto de convergência de pior tempo para processamento das requisições em situações críticas, no intuito de usá-lo como tempo de invisibilidade padrão para as requisições da fila. Com os valores para esses dois parâmetros, capacidade máxima de requisições simultâneas e tempo de invisibilidade para as requisições, será executado o projeto de experimentos. Por fim, serão discutidos os resultados coletados com foco nas duas questões:

1. O DAaaS consegue provisionar recursos dinamicamente de maneira efetiva?
2. O DAaaS é, de fato, tolerante a falhas?

5.1 Carga de Trabalho

Para este trabalho, serão avaliadas as funcionalidades de tradução a partir do texto e da legenda. Como será apresentado na seção 5.3, um dos fatores de variação do projeto de experimentos é a carga de trabalho que pode assumir um valor mínimo ou máximo.

Para requisições de texto utilizamos o sítio do I SENAPES (<http://senapes.lavid.ufpb.br/>)

como base. O SENAPES (Seminário Nacional de Acessibilidade para Pessoas Surdas) acontecerá em março de 2014, será aberto ao público e tem como objetivo principal a apresentação e discussão de experiências no desenvolvimento e implantação de tecnologias assistivas para promoção da acessibilidade para pessoas surdas nas TICs. O número de pessoas surdas que irá acessar o sítio será elevado, e o mesmo será acessível através do DAaaS. Portanto, para requisições de texto, será utilizado o texto principal da página do SENAPES. Como valor mínimo, será utilizada a primeira frase da página principal do SENAPES, que contém 147 caracteres que compõem 24 palavras, e para o valor máximo, o texto completo da página com 2202 caracteres que constituem 324 palavras.

Para requisições de legenda, utilizamos um filme e um seriado como carga de trabalho, uma vez que são modalidades bem difundidas e utilizadas nas TICs. Como valor máximo, foi optado pela legenda de um filme com duração de 2 horas, contendo 30529 caracteres que constituem 3811 palavras. Já para o valor mínimo, utilizamos a legenda de um episódio de um seriado com duração de 21 minutos, contendo 26236 caracteres que compõem 2372 palavras. Ambas as cargas de trabalho são explicitadas no Anexo B.

5.2 Testes Preliminares

O primeiro objetivo dos testes preliminares é definir a capacidade máxima de requisições simultâneas que cada tipo de instância consegue atender. Sabe-se que cada instância consegue atender mais de uma requisição simultânea, portanto, deve ser aproveitada a capacidade máxima dos recursos para proporcionar um melhor QoS ao mesmo tempo em que se reduz os custos financeiros.

Para tolerância a falhas foi concebida uma fila cujos elementos são marcados como visíveis ou invisíveis de acordo com seu estado. Ao entrar na fila, uma requisição é então marcada como visível para que todas as instâncias Ec2 possam processá-las. No momento em que uma instância de processamento consome uma requisição da fila, tal requisição é configurada como invisível para que não seja processada de forma duplicada por outra instância. Contudo, se for detectado algum comportamento estranho no processamento de uma requisição, como, por exemplo, a mesma demorar muito tempo a ser processada, supõe-se que ela falhou. Para tanto, a Fila de Requisições tem um mecanismo de auto-recuperação

que consiste em tornar as requisições novamente visíveis se as mesmas demorarem um tempo razoavelmente elevado para serem processadas.

O segundo objetivo dos testes preliminares é estipular um “pior tempo” de processamento de requisições para cada tipo de instância. Se tais testes não fossem executados, esse tempo seria escolhido de forma completamente aleatória, e poderia ser muito longo, fazendo com que o DAaaS não aproveitasse os seus recursos de forma eficiente, ou poderia ser muito curto, implicando em tornar invisíveis requisições que não falharam com uma frequência elevada.

Apesar de hoje o VLIBRAS possuir um dicionário com cerca de 800 sinais, para estas simulações foi utilizado um dicionário com 384 sinais (quantidade de sinais existentes na época em que a primeira simulação foi executada). Sabe-se que as glosas não encontradas no dicionário de sinais são soletradas por datilologia, o que gera uma sobrecarga de tempo no vídeo final, tornando o tamanho do vídeo de tradução maior. Portanto, os tempos apresentados nas Tabelas 5.3 e 5.6 bem como a qualidade da tradução gerada podem ser melhorados com a evolução do dicionário de sinais.

As configurações de *hardware* das instâncias *m1.small* HD (*Hard Disk*) e *c3.xlarge* SSD (*Solid State Disk*), utilizadas nos experimentos preliminares, são descritas na Tabela 5.1. A intensidade de processamento das mesmas são medidas em termos de vCPUs e ECUs (*Elastic Computing Unit*). vCPU se refere à quantidade de núcleos virtualizados, e ECU é uma medida para facilitar a comparação do poder de processamento entre as instâncias. Cada ECU é definido como o poder computacional de um processador 1.0-1.2 GHz 2007 *Opteron* ou 2007 *Xeon*. Para simplificar, podemos explicitar o processamento da instância *m1.small* como sendo de 1 ECU, e o processamento da instância *c3.xlarge* como sendo de 14 ECUs, i.e., 4 núcleos (vCPUs) de 3.5 ECUs cada.

Tabela 5.1: Configuração de *hardware* das instâncias

Tipo de Instância	Preço	vCPU	ECU	Mem. RAM	Armazenamento (GB)	Desempenho de Rede
<i>m1.small</i>	\$0.06 p/ hora	1	1	1.7GB	1x160	baixo
<i>c3.xlarge</i>	\$0.3 p/ hora	4	14	7.5GB	2x40 SSD	moderado

No intuito de facilitar a compreensão dos valores dos tempos dos experimentos preliminares (Tabelas 5.3 e 5.6), foram realizados alguns testes para identificarmos o tempo de

execução dos principais componentes. Os principais componentes do VLIBRAS são:

1. Extração: o primeiro passo é a extração da legenda. A etapa de extração é responsável por colher segmentos de texto (frases) do arquivo de legenda, e transmiti-las ao Tradutor.
2. Tradução Automática: responsável por converter o texto de entrada, na língua portuguesa, para um conjunto de glosas correspondentes (LIBRAS).
3. Sincronização e Animação: estes componentes recebem as etiquetas de tempo para cada frase traduzida, e as respectivas glosas. Sua principal função é ajustar o tempo que cada sinal deve ser exibido no vídeo de tradução, e escrever esse vídeo final em disco.

É importante ressaltar que as três tarefas são executadas paralelamente, onde cada componente representa uma *thread*. Assim que o Extrator termina de extrair uma frase do arquivo de legenda, ele prontamente inicia a extração das próximas sentenças, independentemente do Tradutor (próximo componente no processo de tradução) ter processado ou não a sentença anterior. Estas frases são processadas pelo Tradutor que por sua vez repassa ao Sincronizador as glosas e as etiquetas de tempo para cada uma delas traduzidas. Portanto, o desempenho de cada *thread* só depende da velocidade da *thread* anterior: o Sincronizador depende do Tradutor que por sua vez depende do Extrator, ou seja, o Sincronizador irá demorar mais que o Tradutor que irá demorar mais que o Extrator.

Para esses testes dos componentes, foi utilizada a mesma legenda dos experimentos preliminares (Tabelas 5.3 e 5.6). Essa legenda é de um filme de 2h e resulta em um vídeo de tradução com 1.6GB de carga. Foram medidos os tempos de tradução desta legenda 10 vezes em três instâncias: *c3.xlarge* com armazenamento de 2x40GB SSD, *c3.xlarge* com armazenamento de 1x160GB HD, *m1.small* com armazenamento de 1x160GB HD. É importante lembrar que é possível utilizar a instância *c3.xlarge* sem montar as partições SSD, e utilizar apenas a partição raiz (HD) para armazenamento. A tabela abaixo contém os tempos das 10 repetições de 1 requisição em cada uma delas.

Conclusões tiradas a partir dos resultados desta Tabela (5.2):

1. Extrator: a média de tempo do Extrator é similar nas instâncias *c3.xlarge* SSD e *c3.xlarge* HD, uma vez que ambas possuem as mesmas capacidades de processamento,

Tabela 5.2: Tempos de processamento dos componentes em três instâncias diferentes

<i>c3.xlarge</i> - SSD			<i>c3.xlarge</i> - HD			<i>m1.small</i> - HD		
Ext e Trad	Sinc e Anim	Tot	Ext e Trad	Sinc e Anim	Tot	Ext e Trad	Sinc e Anim	Tot
4.71	9.93	11.42	5.17	56.23	57.46	22.97	68.66	73.45
5.59	8.80	10.08	4.56	53.83	55.06	21.96	62.88	67.65
5.89	9.59	10.86	5.20	53.83	55.06	22.86	62.45	67.58
4.77	8.78	10.06	4.60	65.44	66.70	24.20	52.29	57.82
5.33	9.26	10.66	5.21	58.03	59.31	24.63	52.87	57.67
6.00	9.58	10.87	4.54	55.43	56.67	25.20	57.21	62.41
4.76	8.90	10.26	5.21	61.64	62.90	24.26	56.55	61.53
5.57	9.54	10.84	5.17	53.63	54.85	24.85	53.67	58.54
5.52	9.45	10.85	5.11	60.63	61.93	25.60	52.44	57.39
5.53	9.46	10.85	5.16	55.63	56.86	23.35	54.79	59.68
Média								
5.37	9.33	10.68	4.99	57.43	58.68	23.99	57.38	62.37

diferindo apenas no disco. Como a máquina *m1.small* HD tem menor poder de processamento, o tempo do seu Extrator é superior aos da *c3.xlarge* SSD e HD.

2. Sincronizador: seu principal trabalho é escrita em disco. Ao comparar os tempos do Sincronizador na máquina *c3.xlarge* SSD para a *c3.xlarge* HD, vemos que, para esse experimento, a máquina com SSD é em média, 6x mais rápida do que a que possui HD, ainda que ambas possuam mesmo poder de processamento e memória. Contudo, ao comparar o tempo do Sincronizador da *c3.xlarge* HD pra *m1.small* HD, podemos perceber que não há diferença, pois a tarefa do Sincronizador é basicamente escrita em disco, e como as duas tem o mesmo tipo de disco, possuem as mesmas velocidades. É importante perceber que por mais que o tempo do Extrator da *m1.small* seja maior do que a *c3.xlarge* HD, ele ainda não se torna gargalo para escrita, ou seja, ele não se torna lento o suficiente a ponto de atrasar ainda mais a escrita. Por mais que o Sincronizador seja lento, ele vai sempre conseguir manter o *buffer* da memória cheio para o (lento) disco escrever, o que nos permite identificar a escrita (componente Sincronizador) como o grande gargalo do VLIBRAS.
3. Tempo total: é possível perceber superioridade de desempenho do disco SSD em re-

lação ao HD, quando comparamos as máquinas *c3.xlarge* SSD versus *c3.xlarge* HD. Adicionalmente, quando comparamos as máquinas com disco HD (*c3.xlarge* HD e *m1.small* HD), vemos que a *c3.xlarge* HD é, em média, 4 segundos mais rápida que a *m1.small* HD, apesar de 4 requisições da *c3.xlarge* HD terem sido mais lentas quando comparadas às *m1.small* HD, devido ao desvio padrão inerente a procedimentos de escrita.

Os experimentos preliminares foram compostos por 11 simulações para dois tipos de instância Ec2, *m1.small* HD e *c3.xlarge* SSD (que serão utilizadas no projeto de experimentos), e para cada tipo de serviço disponibilizado no presente trabalho, texto e legenda. Em cada simulação foram submetidas 1, 2, 4, 6, 10, 15, 20, 25, 30, 40 e 50 requisições simultâneas, e foram medidos o tempo médio (TM), pior tempo (PT), desvio padrão (DP), e quantidade de falhas (F). As Tabelas 5.4, 5.5, 5.7 e 5.8, resumem as 44 simulações (*m1.small*-texto(11), *m1.small*-legenda(11), *c3.xlarge*-texto(11) e *c3.xlarge*-legenda(11)), no intuito de facilitar a análise dos dados de forma generalizada. Os dados completos que resultaram na síntese das 44 simulações encontram-se no Apêndice C.

Conforme mencionado anteriormente, os objetivos dos experimentos preliminares são definir o número máximo de requisições simultâneas que cada tipo de instância consegue atender, e estipular um “pior tempo” de processamento de requisições para cada tipo de instância. Para os experimentos preliminares foi decidido que o limite de falhas aceitáveis seria 10%. Portanto, as requisições simultâneas que obtiveram quantidade de falhas superiores a 10% foram marcadas em vermelho. Para facilitar a escolha, grifamos de cinza as duas linhas candidatas a serem escolhidas para basear a capacidade máxima de requisições simultâneas e o “pior tempo” de processamento de requisições. Para tal, foram escolhidas as que conseguem processar mais requisições simultâneas sem oferecer uma quantidade (ou perigo) de falhas superior a 10%. É importante perceber que, para esses valores, devemos encontrar um ponto de convergência entre as modalidades texto e legenda, pois ambas serão processadas na mesma máquina.

Tabela 5.3: Instância *m1.small* HD

Tabela 5.4: Texto

Nº req.	Tempo Médio	Pior Tempo	Desvio Padrão	Falhas
1	24.22 s	24.86 s	0.47 s	0.00%
2	37.1 s	45.28 s	3.47 s	0.00%
4	67.77 s	78.31 s	4.63 s	0.00%
6	88.7 s	136.69 s	17.23 s	0.00%
10	139.57 s	182.62 s	23.8 s	0.00%
15	192.39 s	225.89 s	24.97 s	0.00%
20	200.59 s	264.62 s	34.44 s	0.00%
25	250.37 s	322.14 s	45.21 s	14.80%
30	272.2 s	415.97 s	91.37 s	0.00%
40	332.73 s	578.55 s	136.91 s	0.00%
50	519.24 s	896.04 s	218.35 s	0.20%

Tabela 5.5: Legenda

Nº req.	Tempo Médio	Pior Tempo	Desvio Padrão	Falhas
1	83.2 s	175.09 s	31.94 s	0.00%
2	281.7 s	330.52 s	33.58 s	5.00%
4	544.28 s	603.63 s	44.66 s	2.50%
6	854.05 s	1175.62 s	143.64 s	3.33%
10	1244.45 s	1516.44 s	133.14 s	1.00%
15	1323.32 s	1668.18 s	251.28 s	2.00%
20	1459.87 s	1929.89 s	275.34 s	11.00%
25	1626.94 s	2148.23 s	277.38 s	26.40%
30	2380.78 s	3534.34 s	601.31 s	22.33%
40	2907.73 s	6306.88 s	1730.94 s	24.00%
50	4642.63 s	8879.32 s	2904.02 s	57.20%

Tabela 5.6: Instância *c3.xlarge* SSD

Tabela 5.7: Texto

Nº req.	Tempo Médio	Pior Tempo	Desvio Padrão	Falhas
1	5.19 s	6.87 s	0.78 s	0.00%
2	5.09 s	6.27 s	0.7 s	0.00%
4	8.3 s	13.53 s	2.37 s	0.00%
6	11.54 s	18.45 s	3.2 s	0.00%
10	18.2 s	29.00 s	5.45 s	0.00%
15	29.27 s	42.03 s	7.14 s	0.00%
20	41.59 s	56.27 s	9.63 s	0.00%
25	43.79 s	69.67 s	12.92 s	0.00%
30	55.67 s	87.78 s	17.32 s	0.00%
40	72.83 s	117.96 s	22.62 s	0.00%
50	88.71 s	139.41 s	26.87 s	0.00%

Tabela 5.8: Legenda

Nº req.	Tempo Médio	Pior Tempo	Desvio Padrão	Falhas
1	23.02 s	25.79 s	2.78 s	0.00%
2	21.81 s	34.27 s	6.38 s	5.00%
4	42.91 s	73.31 s	14.53 s	7.50%
6	66.41 s	90.75 s	16.63 s	6.00%
10	101.5 s	166.63 s	34.7 s	1.00%
15	153.06 s	251.23 s	54.66 s	4.60%
20	212.17 s	336.33 s	62.04 s	0.00%
25	241.1 s	383.77 s	52.11 s	3.60%
30	307.07 s	513.58 s	81.63 s	0.67%
40	368.91 s	478.00 s	49.31 s	3.25%
50	449.74 s	534.53 s	45.68 s	0.60%

Baseado nos resultados expostos nas Tabelas 5.3 e 5.6, foram escolhidos os seguintes valores para o número máximo de requisições simultâneas e pior tempo de processamento (Tabela 5.9).

Tabela 5.9: Valores escolhidos nos experimentos preliminares

Instância	Nº de requisições simultâneas	Tempo de invisibilidade padrão
<i>m1.small</i>	15	1700 s
<i>c3.xlarge</i>	49	550 s

Analisando a Tabela 5.3, é possível observar que o limite da instância *m1.small* para traduções simultâneas a partir da legenda com quantidade de falhas inferior a 10% é o número 15. Como ambas opções de serviço serão processadas na mesma máquina, 15 requisições simultâneas também se torna o limite superior para a tradução de texto, mesmo sabendo que a instância poderia processar mais do que 15 traduções simultâneas de texto. Nesse sentido, foi considerado o pior caso envolvendo a intersecção entre texto e legenda, que foi a possibilidade de uma instância processar 15 traduções simultâneas de legenda. Nesse caso, o pior tempo coletado nos experimentos foi 1668.18 s, que aproximamos para 1700 s.

Sabendo que o gargalo do VLIBRAS é a escrita em disco, um dado importante a ser relatado é o tamanho final dos vídeos de tradução para a carga de trabalho máximo de texto e legenda. Ao traduzir um texto com 2202 caracteres que constituem 324 palavras - nosso limite superior para traduções de texto - o vídeo de tradução gerado tem 122MB de tamanho. Contudo, ao traduzir a legenda de um filme com duração de 2 horas, contendo 30529 caracteres que constituem 3811 palavras - nosso limite superior para traduções de legenda - o vídeo de tradução gerado tem 1.6GB de tamanho. Isto torna claro o motivo pelo qual consideramos como pior caso uma eventualidade em que todas as requisições processadas por uma instância sejam a partir de uma legenda, já que seu vídeo de tradução é 13.11 vezes maior que o vídeo de tradução para texto.

Analisando a Tabela 5.6, vemos que o limite da instância *c3.xlarge* para traduções simultâneas a partir da legenda com quantidade de falhas inferior a 10% é o número 50. Apesar de perceber que a instância é capaz de processar mais de 50 requisições mantendo um percentual de falhas inferior a 10%, temos que limitar este número a 50 ou menos devido à capacidade de armazenamento do disco SSD para a instância *c3.xlarge*. A instância *c3.xlarge* possui

capacidade de armazenamento de $2 \times 40\text{GB} = 80\text{GB}$ SSD, que permite a escrita de 50 vídeos de tradução de legenda ($50 \times 1.6\text{GB} = 80\text{GB}$). Como esse limite está muito justo, foi escolhido o valor 49 como o número máximo de requisições simultâneas. Para 50 requisições simultâneas o pior tempo coletado nos experimentos foi 534.53 s, que foi aproximado para 550 s e usado como limite folgado para o processamento de 49 traduções simultâneas.

Com esses experimentos preliminares, é possível observar que o programa VLIBRAS é passível de falhas, mesmo em condições favoráveis, vide 2/4/6 requisições simultâneas da Tabela 5.5 e a Tabela 5.8 como um todo. Esse é um dos fatores que eleva a necessidade de haver uma política de tolerância a falhas em nível de *software*.

Na próxima seção será descrito o projeto de experimentos e seus respectivos resultados.

5.3 Projeto de Experimentos

O planejamento do experimento foi realizado utilizando-se a técnica de projeto fatorial 2^k . Tal técnica é usada para analisar os efeitos de k fatores, onde todos eles tem dois níveis. Os níveis podem ser quantitativos (nº de requisições simultâneas e injeção de falhas) ou qualitativos (tipo da instância Ec2 e carga de trabalho). Uma repetição completa consiste em 2^k unidades experimentais. Quando o experimento envolve muitos fatores, os projetos fatoriais completo e fracionado podem implicar restrições de tempo, ao passo que o fatorial 2^k por só ter dois níveis para cada fator, tende a resultar em um número plausível de experimentos. Porém, para estimar os erros experimentais e agregar valor ao resultado dos experimentos é preciso replicar os 2^k experimentos. Desse modo, têm-se $2^k r$ observações, e pode-se comparar o grau de variação devido a cada fator, devido a interação entre eles, e devido aos erros experimentais.

Primordialmente, pretende-se avaliar a capacidade de provisionamento dinâmico e de tolerância a falhas. Para tanto, nosso projeto fatorial $2^k r$ contará com os fatores “número de requisições simultâneas” assumindo o valor 10 ou 500, e “injeção de falhas” com valor 0 ou 10%. Adicionalmente, queremos saber a influência que a “carga de trabalho” e o “tipo de instância Ec2” terão sobre o tempo de geração dos vídeos. Nosso projeto fatorial $2^k r$ terá $k = 4$ e $r = 20$. A Tabela 5.10 detalha os fatores e suas variações.

Como nosso k é 4, teremos 16 (2^4) experimentos possíveis entre os fatores, que se-

Tabela 5.10: Fatores do projeto fatorial $2^k r$

Fatores		Mínimo (-1)	Máximo (1)
A	Injeção de falhas	0	10%
B	Nº de req. simultâneas	10	500
C	Carga de trabalho	Leg. (21 min.) e texto (581 <i>chars</i>)	Leg. (2h) e texto (2202 <i>chars</i>)
D	Tipo de Instância Ec2	<i>m1.small</i>	<i>c3.xlarge</i>

rão replicados 20 vezes¹, resultando em 320 experimentos. A Tabela D.1 no apêndice D especifica os 16 experimentos, exibindo o valor que as variáveis A, B, C, e D assumirão, além dos valores para a interação entre estes fatores. Os valores para AB/AC/AD/BC/BD/CD/ABC/ABD/ACD/BCD/ABCD são calculados a partir do produto do seus fatores. A coluna ABC, por exemplo, é resultado da multiplicação dos valores de A, B, e C. A partir desses valores, e da média das replicações, é possível calcular o impacto de cada fator no tempo final, além do grau de importância das interações entre eles.

5.3.1 Execução do Projeto de Experimentos

Para cada teste utilizamos duas instâncias Ec2 *c3.large* como base. Uma instância foi utilizada como origem das requisições, e a outra iniciada com o *Broker* e configurações específicas para cada experimento. Escolhemos a instância *c3.large* pois ela é dotada de 3.75 GB de memória RAM, 2 núcleos de 3.5 ECUs cada (2 vCPUs e 7 ECUs), e desempenho de rede “moderado”.

A capacidade de memória RAM é importante ao *Broker* uma vez que configuramos o uso de memória RAM do servidor *Tomcat 7* para 2048 MB, onde o padrão é apenas 128 MB (Código Fonte E.1 em Apêndice E). Sem essa alteração, o *Broker* lançava exceções do tipo “*java.lang.OutOfMemoryError*”, já que em alguns experimentos o *Broker* cria 500 *threads* simultâneas ao receber as 500 requisições.

O fato da máquina *c3.large* ter sido escolhida também se deve a seu desempenho de rede “moderado”, já que existem instâncias com desempenho classificado “baixo”. Isto é importante para que a rede não seja gargalo para nossos experimentos, não interferindo no resultado final. Neste sentido, também foi preciso reconfigurar a *tag Connector* do arquivo

¹Com 20 replicações conseguimos alcançar um erro experimental aceitável de 11.47%

de configurações do servidor (/var/lib/tomcat7/conf/server.xml) para elevar sua capacidade de atender requisições simultâneas (Código Fonte ?? em Apêndice E).

Para simular as requisições simultâneas utilizamos a ferramenta *ab* - *Apache HTTP server benchmarking* (<http://httpd.apache.org/docs/current/programs/ab.html>). A linha de comando apresentada no Código Fonte E.2, é utilizada para simular o 1º experimento para A=-1, B=1, C=-1 e D=-1 (sem injeção de falhas, com 500 requisições simultâneas, carga de trabalho baixa, instância Ec2 *m1.small*). A Tabela E.1 explica o que cada parâmetro utilizado no *ab* representa. O Código Fonte E.3, no Apêndice E exemplifica a saída gerada pela ferramenta *ab* para o experimento executado na linha de comando do Código Fonte E.2, referente as requisições de legenda. O principal conteúdo coletado é a linha 34 que nos detalha a média de tempo e desvio padrão em milissegundos para aquele experimento.

Injeção de Falhas

Para injetar 10% de falhas referentes as requisições submetidas, inserimos um trecho de código (5.1) no componente VLIBRAS que a cada 10 requisições recebidas, a primeira falha.

Código Fonte 5.1: Injeção de 10% de falhas no VLIBRAS

```

1  /* Injecao de 10% de falhas no atendimento das requisicoes */
2      if (VLibras.mayFail){
3          VLibras.incrementaNumRequisicoesRecebidas();
4          if (VLibras.getNumRequisicoesRecebidas() == 1)
5              return null;
6          else if (VLibras.getNumRequisicoesRecebidas() == 10)
7              VLibras.setNumRequisicoesRecebidas(0);
8      }
```

Os métodos “*VLibras.incrementaNumRequisicoesRecebidas()*”, “*VLibras.getNumRequisicoesRecebidas()*” e “*VLibras.setNumRequisicoesRecebidas(int)*”, são sincronizados, e portanto seguros quanto à concorrência de *threads* simultâneas.

É importante ressaltar que a injeção de falhas acontece no componente VLIBRAS, e não nas requisições em si. Por exemplo, se submetermos 500 requisições, não necessariamente teremos 50 falhas, já que as máquinas serão criadas com o decorrer do experimento por meio do escalonamento. As falhas, portanto, estão diretamente relacionadas ao escalonamento e quantidade de instâncias para processar a fila. Se uma instância processar apenas 11 requisições, devido à concorrência com outras instâncias para consumir a fila, 2 falharão,

representando 18% de falhas ao invés de 10%.

5.4 Resultados

A Tabela 5.11 apresenta a média de tempo das requisições para os 16 experimentos replicados 20 vezes. Os dados completos que resultaram na síntese dos 16 experimentos com 20 replicações encontram-se no Apêndice D.

Tabela 5.11: Projeto de Experimentos - Resultados

Exp	A	B	C	D	Média	Replicações																				Falhas	
						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	VLBRAS	DAaaS
1	-1	-1	-1	-1	202.3	402	246	198	184	263	175	166	166	251	201	160	170	172	170	139	238	138	145	307	155	1	0
2	-1	-1	-1	1	102.1	61	40	38	37	265	147	202	207	150	37	38	35	146	91	151	37	146	33	149	34	16	0
3	-1	-1	1	-1	511.1	1302	688	683	691	504	383	414	676	720	508	432	358	340	367	391	309	328	309	361	458	0	0
4	-1	-1	1	1	119.1	88	179	139	94	84	136	184	133	86	86	88	137	171	137	80	131	96	106	95	129	12	0
5	-1	1	-1	-1	938.4	926	1054	905	849	952	919	911	941	868	1127	969	710	1027	1076	978	855	825	922	967	986	17	0
6	-1	1	-1	1	594.0	497	506	562	553	570	813	581	731	580	619	479	614	559	623	623	535	806	559	472	599	149	0
7	-1	1	1	-1	1254.2	1433	1229	1125	1379	1198	1480	1071	1186	997	923	1313	1670	1719	1067	1297	1140	1310	1017	954	1579	32	0
8	-1	1	1	1	1021.3	924	1111	941	970	993	1029	1014	916	944	1004	992	1179	1045	1040	1097	967	1153	1020	1014	1072	152	0
9	1	-1	-1	-1	343.5	314	447	331	492	290	155	311	549	333	170	458	331	513	307	313	145	458	324	326	305	21	0
10	1	-1	-1	1	173.0	157	425	150	259	265	34	150	104	91	372	148	145	144	198	248	89	101	89	259	32	32	0
11	1	-1	1	-1	580.3	442	817	558	762	413	713	585	371	705	517	730	488	532	857	423	500	548	515	574	559	21	0
12	1	-1	1	1	231.7	195	180	130	181	239	242	186	462	123	238	186	176	237	354	243	183	244	71	240	526	36	0
13	1	1	-1	-1	1144.6	1350	1197	1218	1043	1094	1105	897	1149	1200	1207	1372	1194	1120	1055	1115	1219	1158	1124	1001	1074	1177	0
14	1	1	-1	1	654.4	596	600	598	559	547	571	560	641	665	586	720	724	722	683	671	695	684	765	789	710	1262	0
15	1	1	1	-1	1558.6	1900	1821	1746	923	830	1125	1745	1662	1622	1577	1560	1490	1767	1447	1750	1768	1656	1570	1675	1538	1283	0
16	1	1	1	1	1095.8	1077	1045	1149	303	901	1388	2055	1149	1564	901	1373	886	975	943	784	799	1286	1462	799	1077	1112	0

O projeto de experimentos resultou em 81420 vídeos de tradução, que foram armazenados no S3 ocupando 49.49 TB. O *Apache Benchmarking* não conseguiu submeter 0.22% das requisições (180 de um total de 81600). Das 81420 requisições submetidas o VLIBRAS falhou 5323 vezes, 6.53% das requisições. Lembrando que apenas 8 das 16 combinações de fatores injetaram 10% de falhas, então deveríamos obter 5%, contudo, como já explicado na seção 5.3.1, de acordo com o escalonamento a porcentagem de falhas injetada tende a ser pouco maior que 10%. Adicionalmente, nesse número de falhas estão inclusas as falhas inerentes (não injetadas) ao VLIBRAS. Por último, a API DAaaS não apresentou nenhuma falha. Todas as requisições submetidas ao DAaaS foram atendidas com sucesso, retornando a *url* do vídeo de tradução no S3, embora existissem requisições que demorassem cerca de 2 horas para serem respondidas.

A Tabela 5.12 exhibe o impacto de cada fator, interação entre eles, e dos erros experimentais, na variação do tempo de tradução dos vídeos. Considerando os fatores isolados, é possível observar que os fatores B, D e C são, respectivamente, os que mais implicam na variação do tempo final. Com relação a interação entre dois fatores, a interação BC é a que possui mais relevância, ainda que seja pouca comparada a B, D, e C. Já para a interação entre três fatores, constatamos que a interação BCD é a que possui maior peso, mesmo que quase irrelevante quando comparada a B, D, e C. A seguir, vamos analisar os seguintes fatores que mais influenciam a variação do tempo de resposta das requisições: B, D e C. Também analisaremos o fator A para discutir os efeitos das falhas injetadas no DAaaS.

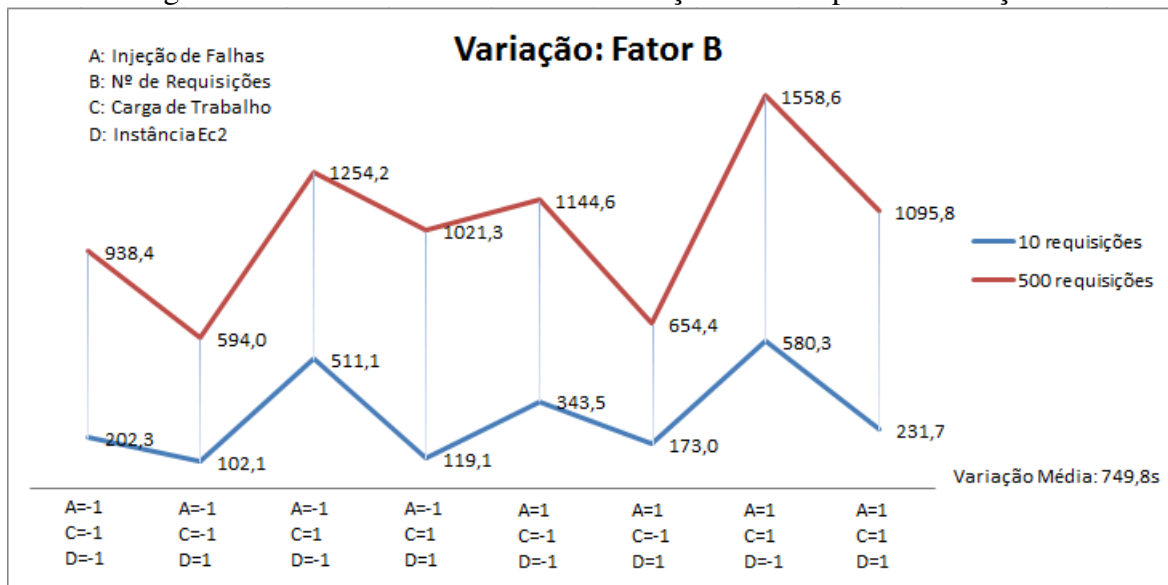
Tabela 5.12: Projeto de Experimentos - Influência dos Fatores na Variação dos Resultados Coletados

A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD	
QA	QB	QC	QD	QAB	QAC	QAD	QBC	QBD	QCD	QABC	QABD	QACD	QBCD	QABCD	
64.96	374.88	138.72	-158.86	15.71	5.13	-25.16	61.08	-32.44	-20.70	8.91	-21.80	1.86	38.06	-12.36	
SSA	SSB	SSC	SSD	SSAB	SSAC	SSAD	SSBC	SSBD	SSDC	SSABC	SSABD	SSACD	SSBCD	SSABCD	SSE
1.89%	62.99%	8.63%	11.31%	0.11%	0.01%	0.28%	1.67%	0.47%	0.19%	0.04%	0.21%	0.00%	0.65%	0.07%	11.47%

Os resultados do projeto de experimentos nos permitem concluir que o fator B - número de requisições simultâneas - é responsável por 62.99% da variação dos tempos de tradução, sendo o fator mais impactante. Observando a Figura 5.1, é possível observar que as curvas seguem um padrão bem definido, causado por uma variação média de 749.8 segundos.

A partir dessa figura também é possível notar a capacidade de provisionamento dinâmico de recursos, ao analisar o caso com maior variação: penúltima coluna (A=1, C=1, D=-1). Nesse caso, o DAaaS atende 10 requisições em 580.3 segundos, portanto, se o mesmo não fosse dotado de uma política de escalonamento, e o tempo de resposta para as requisições simultâneas aumentasse linearmente com sua quantidade, 500 requisições deveriam ser atendidas em cerca de 29015 segundos (50×580.3). Contudo, para essa situação específica (A=1, C=1, D=-1) o DAaaS conseguiu atender as 500 requisições em 1558.6 segundos, nos levando a crer que a capacidade de provisionamento dinâmico do DAaaS o torna aproximadamente 18.6 vezes mais rápido ($29015/1558.6$).

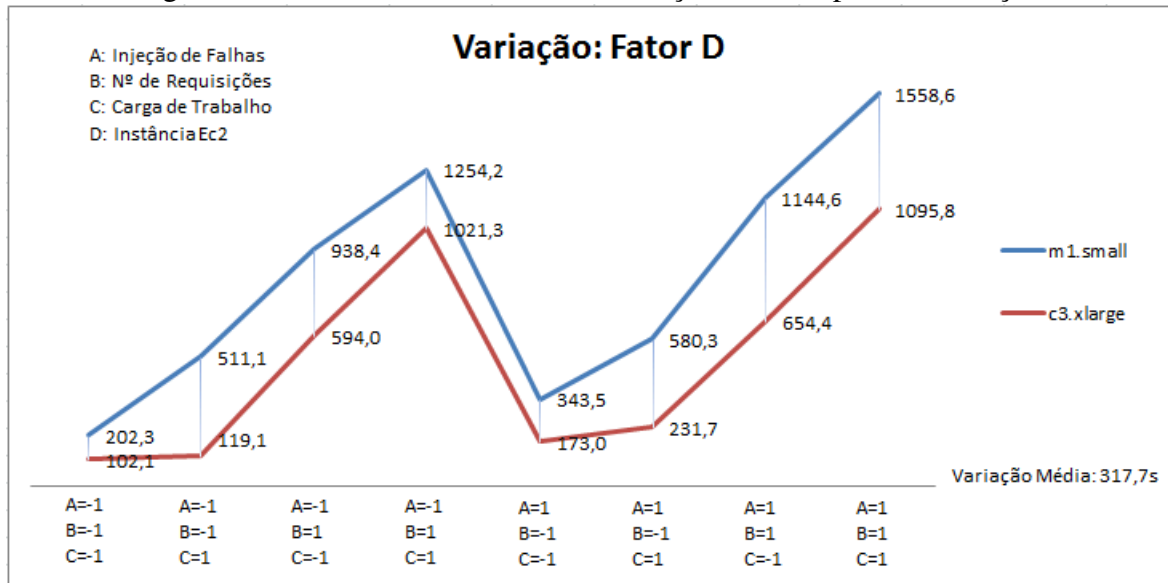
Figura 5.1: Influência do Fator B na Variação dos Tempos de Tradução



Outro ponto de investigação no projeto de experimentos era saber dentre os tipos de instância *m1.small* e *c3.xlarge* qual apresentaria melhor custo benefício em relação ao tempo de tradução e dinheiro investido. A instância *m1.small* custa \$0.06 por hora, enquanto a *c3.xlarge* custa \$0.30 por hora. A partir dos dados da Figura 5.2 criamos um gráfico com

o custo financeiro médio² das instâncias Ec2 nos 16 experimentos. Concluímos que o tipo de instância a ser escolhido depende primordialmente da quantidade de requisições a qual o DAaaS será submetido. Observando a Figura 5.3, é possível notar que para os experimentos com 10 requisições a máquina *m1.small* foi mais barata, mas para 500 requisições o tipo *c3.xlarge* foi \$0.18 menos custoso. Portanto, o uso do tipo *c3.xlarge* passa a se tornar interessante apenas quando o DAaaS é submetido a um grande número de requisições.

Figura 5.2: Influência do Fator D na Variação dos Tempos de Tradução



Na Figura 5.4, é possível observar que o peso da carga de trabalho é o terceiro maior fator impactante na variação do tempo de resposta. Porém, o fator C é pouco relevante quando o comparamos com o fator B.

Analisando a variação proporcionada pelo fator A podemos constatar que a injeção de falhas causa pouco impacto na variação do tempo de resposta das requisições. Adicionalmente, o fato de que 5323 das 81420 requisições submetidas ao VLIBRAS falharam mas o DAaaS conseguiu atender todas elas, nos remete a bons indícios que a solução proposta é tolerante a falhas.

²Custo financeiro médio pois o calculamos a partir da média dos tempos finais dos experimentos

Figura 5.3: Custo das instâncias Ec2 nos 16 experimentos

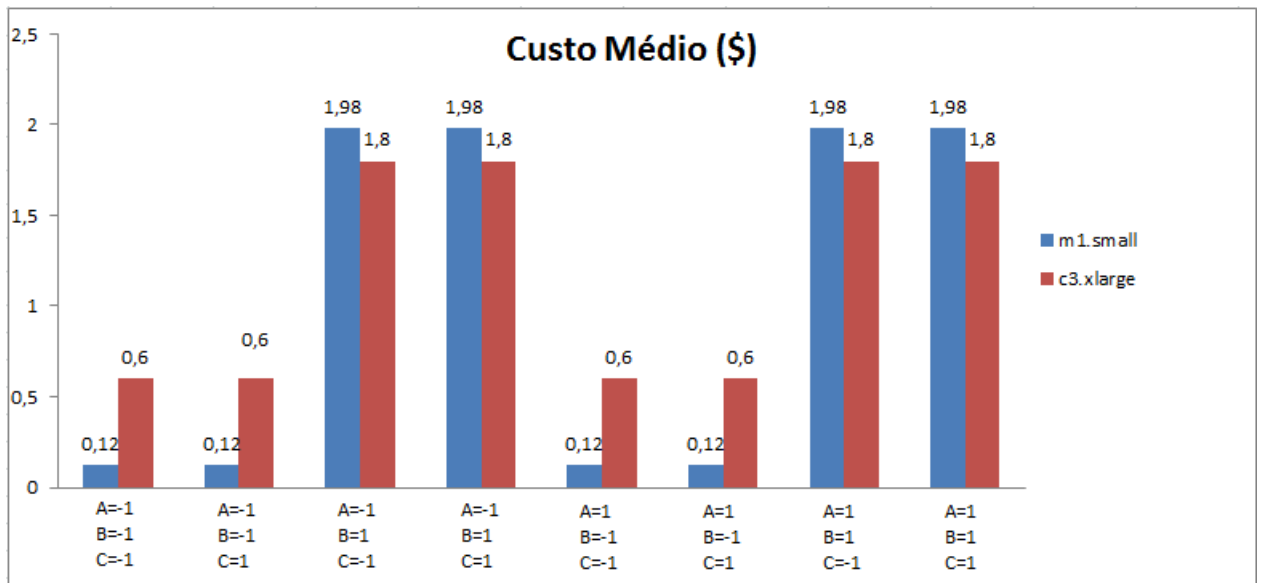
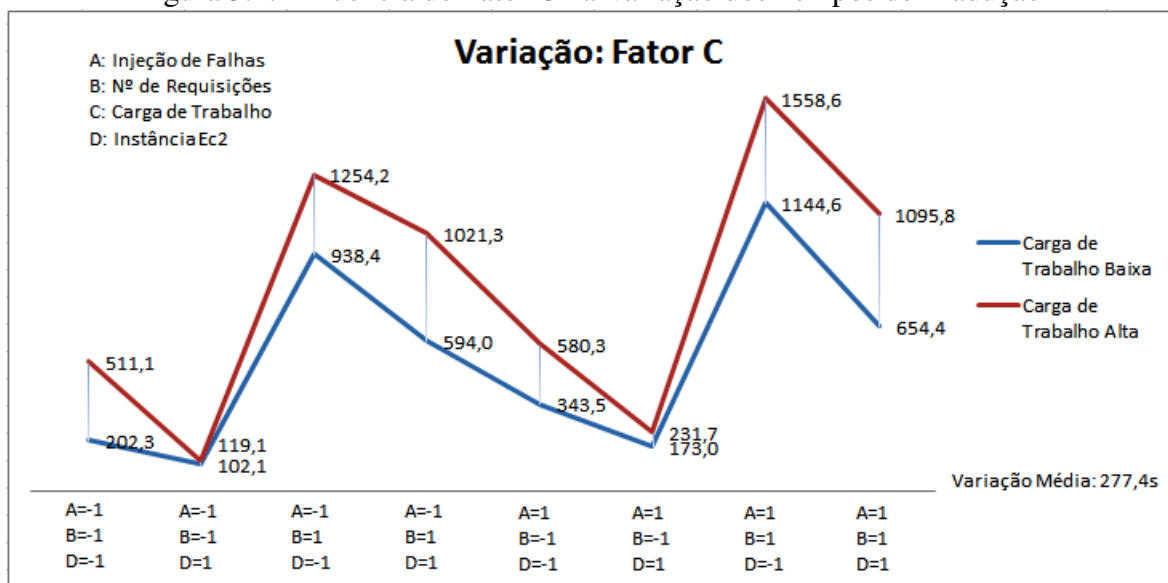


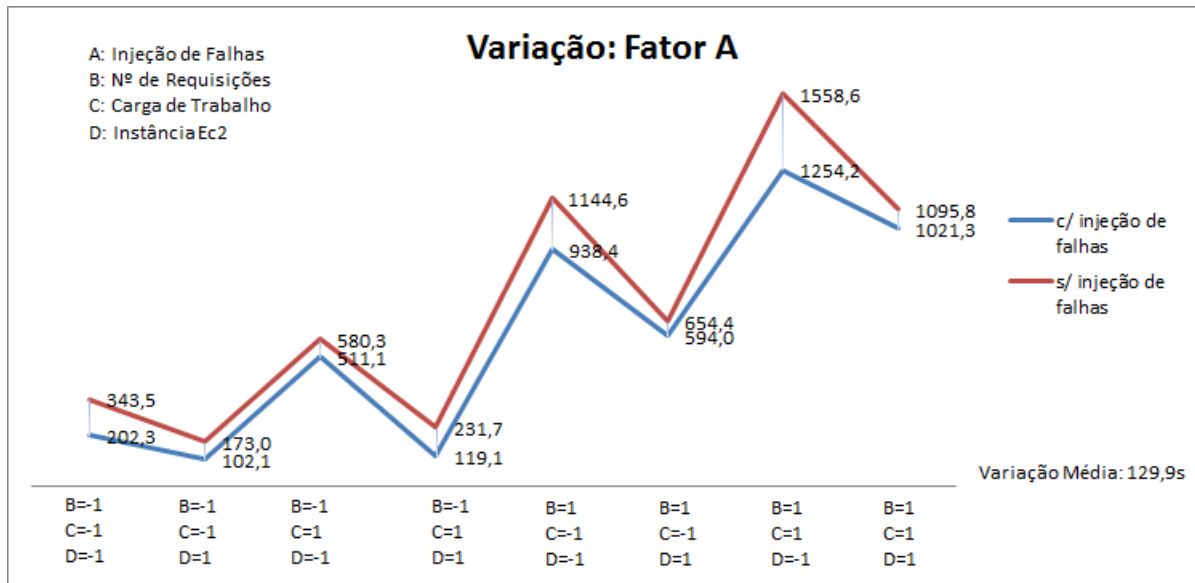
Figura 5.4: Influência do Fator C na Variação dos Tempos de Tradução



5.5 Considerações

Nesse Capítulo detalhamos o projeto de experimentos com objetivo validar a arquitetura proposta através da avaliação da capacidade de provisionamento dinâmico de recursos e de tolerância a falhas. Adicionalmente, também investigamos a influência do tipo de instância Ec2 a ser utilizada e peso da carga de trabalho. Os resultados do projeto de experimentos

Figura 5.5: Influência do Fator A na Variação dos Tempos de Tradução



nos permitiu concluir que o número de requisições simultâneas é o fator mais impactante (62.99%), seguido pelo tipo de instância (11.31%) e pela peso da carga de trabalho (8.63%). Também constatamos que a capacidade de provisionamento dinâmico de recursos torna o sistema aproximadamente 18.6 vezes mais rápido. Ao analisarmos a influência do tipo de instâncias Ec2 na variação dos tempos, percebemos que o tipo *c3.xlarge* passa a ser menos custoso (além de ser mais rápido) que o *m1.small* quando o DAaaS é submetido a um grande número de requisições. Além disso, notamos que o peso da carga de trabalho na variação do tempo de tradução é pouco relevante quando o comparamos com o fator B. Por fim, a variação proporcionada pela injeção de falhas causa pouco impacto na variação do tempo de resposta das requisições. E ao lembrarmos o fato de que 5323 das 81420 requisições submetidas ao VLIBRAS falharam, mas o DAaaS conseguiu atender todas elas, nos remete a bons indícios de que o DAaaS é tolerante a falhas. Contudo, embora o DAaaS tenha se recuperado de todas as falhas, o componente *Broker* representa um ponto crítico de falhas, pois se o mesmo falhar todo o serviço DAaaS fica comprometido.

No próximo Capítulo (6) serão apresentadas as considerações finais, fazendo um balanço dos resultados alcançados, comentando a validação da API por meio da utilização por dois clientes, e trabalhos futuros.

Capítulo 6

Considerações Finais

O presente trabalho propõe a disponibilização do sistema de tradução automática VLIBRAS por meio de uma API a ser implantada em uma arquitetura escalável e tolerante a falhas. Para prover recursos dinamicamente utilizamos o paradigma de computação em nuvem, e implantamos a arquitetura no provedor de infraestrutura AWS.

Os objetivos do presente trabalho foram alcançados: concebemos uma arquitetura escalável e tolerante a falhas para um sistema de processamento multimídia, e a validamos por meio de um projeto de experimentos. Com o projeto de experimentos realizado constatamos que o DAaaS consegue provisionar recursos dinamicamente de maneira efetiva e é, de fato, tolerante a falhas. Também validamos o DAaaS quanto a sua utilização, onde dois clientes implementaram a API e estão atualmente a utilizando para tornar conteúdos acessíveis.

Embora todas as falhas acontecidas no VLIBRAS tenham sido recuperadas, o serviço ainda apresenta um ponto crítico passível de falhas que futuramente deve ser corrigido, o *Broker*. Se, porventura, o *Broker* falhar, a API DAaaS para de funcionar. Um modo rápido e fácil de consertar essa vulnerabilidade no provedor de IaaS AWS, é utilizar o *AutoScaling* aliado ao *Elastic Load Balancing - ELB*. Para tal, usaríamos o *AutoScaling* com um “gatilho” para detectar sempre que não houver instância com a AMI do *Broker* ativa, e configuraríamos uma política de escalonamento para criar uma nova instância com a AMI do *Broker* cada vez que o gatilho for disparado. O ELB seria utilizado para redirecionar as requisições para o IP da instância recém criada. Outra forma de utilização seria utilizar o *AutoScaling* e o ELB para manter sempre as duas instâncias do *Broker* ativas, o que aumentaria a disponibilidade do *Broker* mas também elevaria os gastos de forma desnecessária.

Como resultado inicial do trabalho obteve-se a aceitação do artigo intitulado “*Accessibility as a Service: Augmenting Multimedia Content with Sign Language Video Tracks*” que descreve a tese de doutorado de Tiago Maritan Ugulino de Araújo e que será publicado no *Journal of Research and Practice in Information Technology*. Como resultado parcial, também tivemos um artigo aceito no *14'th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'13)*, intitulado *A Scalable and Fault Tolerant Architecture to Provide Deaf Accessibility as a Service*. Pretende-se publicar alguns trabalhos com o resultado final da pesquisa.

Como trabalhos futuros, fica a correção da vulnerabilidade no *Broker*, a implementação de um mecanismo de autenticação para a API, e a incorporação das demais funcionalidades do VLIBRAS: tradução a partir *closed caption*, do áudio, e a incorporação da funcionalidade de mixagem do vídeo de tradução ao vídeo principal. Outra funcionalidade interessante seria a tradução ao vivo para fluxos de vídeos e canais de emissoras de TV.

Bibliografia

ALMEIDA, E. de. *Leitura e surdez: um estudo com adultos não oralizados*. [S.l.]: Revinter, 2000. ISBN 9788573093896. 8

AMORIM, M. L. C. de et al. Rybenátv: Solução para acessibilidade de surdos para tvdigital. *XVI Simpósio Brasileiro de Sistemas Multimídia e Web*, p. 243–248, 2010. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/webmedia/2010/31_webmi_c.pdf. 34

ARAÚJO, T. M. U. de. *Uma Solução para Geração Automática de Trilhas em Língua Brasileira de Sinais em Conteúdos Multimídia*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, Natal, Brasil, 2012. 9, 10, 13, 37

ARMBRUST, M. et al. A view of cloud computing. *Commun. ACM*, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>. 18, 20, 21, 22, 24, 26

AWS, A. W. S. *Amazon Elastic Compute Cloud (Amazon Ec2)*. 2013. Disponível em: <http://aws.amazon.com/pt/Ec2/>. 29

AWS, A. W. S. *Amazon Simple Queue Service (Amazon SQS)*. 2013. Disponível em: <http://aws.amazon.com/pt/sqs/>. 31

AWS, A. W. S. *Amazon Simple Storage Service (Amazon S3)*. 2013. Disponível em: <http://aws.amazon.com/pt/s3/>. 31

AWS, A. W. S. *Amazon SimpleDB (beta)*. 2013. Disponível em: <http://aws.amazon.com/pt/simplifiedb/>. 31

AWS, A. W. S. *Auto Scaling*. 2013. Disponível em: <http://aws.amazon.com/pt/autoscaling/>. 30

AWS, A. W. S. *Elastic Load Balancing*. 2013. Disponível em: <http://aws.amazon.com/pt/elasticloadbalancing/>. 29

AWS, A. W. S. *Produtos e serviços por região*. 2013. Disponível em: <http://aws.amazon.com/pt/about-aws/globalinfrastructure/regional-product-services/>. 30

BALA, A.; CHANA, I. Fault tolerance-challenges, techniques and implementation in cloud computing. *International Journal of Computer Science Issues*, v. 9, n. 1, p. 288–293, 2012. ISSN 16940784. 41, 42

BAPTISTA, F. *F-Libras - Ambiente Integrado De Ensino-Aprendizagem Para Língua Brasileira De Sinais*. Marília: [s.n.], 2007. 35

BARCELAR, R. R. *Notas de Aulas - Fundamentos de Sistemas Distribuídos*. 2013. Disponível em: http://www.ricardobarcelar.com.br/aulas/sd/2-fundamentos_sd.pdf. 14, 15, 17

BRITO, L. *Por uma gramática de línguas de sinais*. Rio de Janeiro, Brasil: Tempo Brasileiro, 1995. ISBN 9788528200690. 8, 9, 10

BUTTUSSI, F.; CHITTARO, L.; COPPO, M. Using web3d technologies for visualization and search of signs in an international sign language dictionary. In: *Proceedings of the twelfth international conference on 3D Web technology*. New York, NY, USA: ACM, 2007. (Web3D '07), p. 61–70. ISBN 978-1-59593-652-3. Disponível em: <http://doi.acm.org/10.1145/1229390.1229401>. 9

BUYA, R. et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, v. 25, n. 6, p. 599 – 616, 2009. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X08001957>. 18

CORADINE, L. C. et al. Interpretação com busca de palavras, expressões e pequenas frases em português para a libras, na forma gestual animada (etapa dois do falibras). In: *II Fórum de Informática aplicada a Pessoas Portadoras de Necessidades Especiais (III Congresso Brasileiro de Computação)*. Itajaí, SC, de. [S.l.: s.n.], 2003. v. 25, p. 1558–1569. 35

COULOURIS, G. *Distributed Systems: Concepts and Design, 4/e*. [S.l.]: Pearson Education, 2009. ISBN 9788131718407. 13, 14

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Distributed systems: concepts and design*. [S.l.]: Addison-Wesley, 2001. (International computer science series). ISBN 9780201619188. 15, 16

DGE, D. de G. E. *Cartilha Técnica: Recomendações de Acessibilidade para a Construção e Adaptação de Conteúdos do Governo Brasileiro na Internet. Documento de Referência: Versão 2.0*. 2005. Disponível em: <http://www.governoeletronico.gov.br/biblioteca/arquivos/e-mag-versao-2.0/view>. 3

DGE, D. de G. E. *Modelo de Acessibilidade em Governo Eletrônico. Documento de Referência: Versão 3.0*. 2011. Disponível em: <http://emag.governoeletronico.gov.br/emag/emag-3.pdf>. 3, 4

DORR, B. J.; JORDAN, P. W.; BENOIT, J. W. *A Survey of Current Paradigms in Machine Translation*. Advances in Computers, 1999. 1-68 p. Disponível em: <http://www.umiacs.umd.edu/users/bonnie/Publications/newai98.pdf>. 10

FERREIRA, F. L. S. *Provendo suporte a Língua de Sinais em middlewares compatíveis com a ITU J.200*. João Pessoa, Brasil: [s.n.], 2012. 13, 37

- FRANCO, N. de M.; BRITO, P. H. da S. Projeto de uma interface inclusiva para deficientes auditivos: O caso do sistema falibras-web. *EDUCTE: Revista Científica do Instituto Federal de Alagoas*, v. 1, n. 2, p. 115–129, 2011. Disponível em: <http://kentron.ifal.edu.br/index.php/educte/article/view/39/64>. 35
- GÓES, M. de. *Linguagem, surdez e educação*. [S.l.]: Autores Associados, 1996. (Coleção Educação contemporânea). ISBN 9788585701208. 1
- GOMES, R. C.; GÓES, A. R. S. E-acessibilidade para Surdos. *Revista Brasileira de Tradução Visual*, v. 7, n. 7, 2011. ISSN 2176-9656. 1, 4
- GROSSMAN, R. The case for cloud computing. *IT Professional*, v. 11, n. 2, p. 23–27, 2009. ISSN 1520-9202. 22, 25, 26
- HUI, C. L. W.; YANG, Y. Mediacloud: a new paradigm of multimedia computing. *KSII Transactions on Internet and Information Systems*, p. 1153–1170, 2012. Disponível em: <http://go.galegroup.com.ez15.periodicos.capes.gov.br/ps/i.do?id=GALE%7CA294195163&v=2.1&u=capes58&it=r&p=AONE&sw=w>. 45, 46, 47
- IBGE, I. B. de Geografia e E. *Censo demográfico 2010: Características gerais da população, religião e pessoas com deficiência*. [S.l.], 2010. Disponível em: ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/tab1_3.pdf. 2
- ICTS, I. C. de Pesquisa e Desenvolvimento em Tecnologia de S. *Rybená Acessibilidade*. 2013. Disponível em: <http://www.grupoicts.com.br/?pg=paginas&area=acessibilidade>. 34
- International Standards Organization. *Basic Reference Model of Open Distributed Processing, Part 1: Overview and guide to use*. 1992. 17
- JANUÁRIO, G.; KOGA, M. L.; LEITE, L. *POLI-LIBRAS: Um Tradutor de Português para Libras*. 2010. Disponível em: <http://www.polilibras.com.br/documentos/monografia.pdf?attredirects=0&d=1>. 36
- JANUÁRIO, G.; KOGA, M. L.; LEITE, L. *Projeto POLI-LIBRAS*. 2013. Disponível em: <http://www.polilibras.com.br/home>. 36
- Lei número 10.098. *Lei número 10.098 de 19 de dezembro de 2000*. 2000. Disponível em: http://www.planalto.gov.br/ccivil_03/Leis/L10098.htm. 2
- LIRA, G. Projeto tlibras-tradutor português x libras. *Acessibilidade Brasil (http://www.acessibilidade.com.br)*, 2003. 35
- LIRA, G. *O impacto da tecnologia na educação e inclusão social da pessoa portadora de deficiência auditiva: Tradutor digital português x Língua brasileira de sinais–Tlibras*. 2009. 36
- MEIRELLES, V.; GALVÃO, A. S. Uma análise da coesão textual e da estrutura narrativa em textos escritos por adolescentes surdos. *Estudos de Psicologia*, v. 9, n. 1, p. 131–144, 2004. ISSN 1413-294X. 8, 9

MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. [S.l.], 2011. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. 19, 20, 21, 22, 23, 25, 26

MICHAEL, M. et al. Scale-up x scale-out: A case study using nutch/lucene. In: *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. [S.l.: s.n.], 2007. p. 1–8. 15

MOREIRA, J. R. et al. Rumo a um sistema de tradução português-libras. *XXII Simpósio Brasileiro de Informática na Educação - XVII Workshop de Informática na Educação*, p. 1543–1552, 2011. Disponível em: <http://ceie-sbc.tempsite.ws/pub/index.php/wie/article/view/1985/1744>. 34

PIGATTO, D. F. *Estudo e implementação de uma solução de softwares aplicativo utilizando computação nas nuvens*. 2009. Disponível em: <http://www.slideshare.net/dhannyel/trabalho-de-concluso-de-curso-de-graduao>. 25, 31

PIVETTA, E. M.; ULBRICHT, V.; SAVI, R. Tradutores automáticos da linguagem português oral e escrita para uma linguagem visual-espacial da língua brasileira de sinais. V *CONAHPA - Congresso Nacional de Ambientes Hipermídia para Aprendizagem*, 2011. xvi, 34, 36, 38

PROATIVA, S. . N. *ProDeaf*. 2013. Disponível em: <http://www.prodeaf.net/>. 36

RADHAKRISHNAN, G. Adaptive application scaling for improving fault-tolerance and availability in the cloud. *Bell Labs Technical Journal*, Wiley Subscription Services, Inc., A Wiley Company, v. 17, n. 2, p. 5–14, 2012. ISSN 1538-7305. Disponível em: <http://dx.doi.org/10.1002/bltj.21540>. 39, 40

RUSSELL, S.; NORVIG, P. *Inteligência artificial*. [S.l.]: CAMPUS - RJ, 2004. ISBN 9788535211771. 1

SAN-SEGUNDO, R. et al. A spanish speech to sign language translation system for assisting deaf-mute people. In: *Proceedings of Interspeech*. Pittsburgh, EUA: [s.n.], 2006. p. 1399–1402. 9

SILVA, D. A. N. dos S. *Uma linguagem expansível para descrição da Língua Brasileira de Sinais*. João Pessoa, Brasil: [s.n.], 2012. 9, 13

SINTRA, S. N. dos T. *Valores de Referência Praticados a Partir de Janeiro de 2013*. 2013. Disponível em: <http://www.sintra.org.br/site/?p=c&pag=precos>. 4

SOUSA, F. R. C. *Notas de Aulas - Introdução a Computação em Nuvem*. 2013. Disponível em: <https://speakerdeck.com/flaviosousa/introducao-a-computacao-em-nuvem>. xiv, 20, 21, 24

STOKOE, W. C. Sign Language Structure. *Annual Review of Anthropology*, Annual Reviews, v. 9, p. 365–390, 1980. 8, 9

STUMPF, M. R. Língua de Sinais: Escrita dos Surdos na Internet. *Proceedings of V Congresso Ibero-Americano de Informática na Educação*, p. 1–8, 2000. 3

TANENBAUM, A.; STEEN, M. van. *Sistemas distribuídos: princípios e paradgmas*. [S.l.]: Pearson Prentice Hall, 2007. ISBN 9788576051428. 13, 14, 15, 16, 17

WAUTERS, L. *Reading comprehension in deaf children: the impact of the mode of acquisition of word meanings*. Tese (Doutorado) — Radboud University, Nijmegen, Holanda, 2005. Disponível em: <http://repository.ubn.ru.nl/handle/2066/19575>. 3

WHO, W. H. O. *Deafness and hearing loss*. 2013. Disponível em: <http://www.who.int/mediacentre/factsheets/fs300/en/>. 2

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, Springer-Verlag, v. 1, n. 1, p. 7–18, 2010. ISSN 1867-4828. Disponível em: <http://dx.doi.org/10.1007/s13174-010-0007-6>. 18, 22, 24, 25, 26, 48

ZHU, W. et al. Multimedia cloud computing. *Signal Processing Magazine, IEEE*, v. 28, n. 3, p. 59–69, 2011. ISSN 1053-5888. xiv, 43, 44, 45

Apêndice A

API DAaaS - Especificação JSON

Código Fonte A.1: Exemplo de JSON para requisições de tradução de texto

```
1 {  
2     "fromText":true,  
3     "text":"A casa de Eduardo foi restaurada!"  
4 }
```

Código Fonte A.2: Exemplo de JSON para requisições de tradução de legenda

```
1 {  
2     "fromSubtitles":true,  
3     "urlSubtitles":"www.legendas.com/legenda.srt"  
4 }
```

Código Fonte A.3: Exemplo de JSON para requisições de tradução do áudio

```
1 {  
2     "fromAudio":true,  
3     "urlAudio":"www.filmes.com/bope2.wav"  
4 }
```

Código Fonte A.4: Exemplo de JSON para requisições de tradução do áudio do vídeo

```
1 {  
2     "fromAudioFromVideo":true,
```

```
3     "urlVideo": "www.filmes.com/bope2.ts"
4 }
```

Código Fonte A.5: Exemplo de JSON para requisições de tradução do *closed caption*

```
1 {
2     "fromClosedCaption": true,
3     "urlVideo": "www.filmes.com/bope.flv"
4 }
```

Código Fonte A.6: Exemplo de JSON para requisições de tradução de legenda com mixagem do resultado com um vídeo

```
1 {
2     "fromSubtitles": true,
3     "urlSubtitles": "www.legendas.com/legenda.srt",
4     "urlVideo": "www.filmes.com/bope.flv",
5     "options": {
6         "size": "BIG",
7         "position": "TOP_RIGHT",
8         "transparency": false
9     }
10 }
```

Código Fonte A.7: Exemplo de JSON para requisições de tradução do áudio do vídeo com mixagem do resultado com o vídeo

```
1 {
2     "fromAudioFromVideo": true,
3     "urlVideo": "www.filmes.com/bope2.ts",
4     "options": {
5         "size": "MEDIUM",
6         "position": "BOTTOM_LEFT",
7         "transparency": true
8     }
9 }
```

```
8     }  
9 }
```

Código Fonte A.8: Exemplo de JSON para requisições de tradução do *closed caption* do vídeo com mixagem do resultado com o vídeo

```
1 {  
2     "fromClosedCaption":true,  
3     "urlVideo":"www.filmes.com/bope.flv",  
4     "options":{  
5         "size":"SMALL",  
6         "position":"BOTTOM_RIGHT",  
7         "transparency":true  
8     }  
9 }
```

Apêndice B

Carga de Trabalho dos experimentos

Texto - valor mínimo:

“I SENAPES - Seminário Nacional de Acessibilidade para Pessoas Surdas será realizados nos dias 3 e 4 de dezembro, no auditório da Reitoria, Campus I”

Texto - valor Máximo:

“As pesquisas e tecnologias de última geração, desenvolvidas pela UFPB e outras instituições e centros de pesquisa e que permitem maior acessibilidade de pessoas surdas à TV Digital, Web e Cinema Digital, serão apresentadas pela primeira vez aos paraibanos, durante o I Seminário Nacional de Acessibilidade para Pessoas Surdas (I SENAPES). O evento será promovido, na UFPB, em João Pessoa, pelo Núcleo de Pesquisa e Extensão Lavid e a Secretaria Nacional de Promoção dos Direitos da Pessoa com Deficiência (SNPD), órgão da Secretaria de Direitos Humanos da Presidência da República.

O SENAPES acontecerá dias 3 e 4 de dezembro, no auditório da Reitoria da UFPB, é aberto ao público e tem como objetivo principal a apresentação e discussão de experiências no desenvolvimento e implantação de tecnologias assistivas para promoção da acessibilidade para pessoas surdas em sistemas e tecnologias da informação e comunicação.

A temática é “Acessibilidade para Pessoas Surdas em Sistemas e Tecnologias da Informação e Comunicação” e, segundo Tiago Maritan, coordenador do evento

e professor do Centro de Informática, a iniciativa reunirá especialistas no tema de várias regiões do país, também a comunidade de deficientes auditivos do Estado, além de pesquisadores, gestores e técnicos dos quadros da administração municipal, estadual e federal que atuam na promoção de políticas de assistência a surdos.

Nos dois dias do SENAPES, o público poderá conhecer de perto tecnologias assistivas desenvolvidas pela equipe do Núcleo Lavid, a exemplo do LibrasTV e CineLibras, voltadas para cidadãos com dificuldades de audição. Iniciativas, dessa natureza, em desenvolvimento por outras universidades, também serão apresentadas, dentre elas os projetos Vlibras, LinkLibras, Prodeaf, Rybená e HandTalk.

Maritan disse que os participantes irão discutir algumas políticas e ações para promoção da acessibilidade para pessoas surdas, a exemplo da proposta de unificação nacional da Língua Brasileira de Sinais (LIBRAS); a convergência para um vocabulário mais uniforme, que tem como objetivo facilitar a comunicação em nível nacional; as políticas públicas federais, e a inclusão da pessoa surda na sociedade brasileira.”

Legenda - valor mínimo:

Episódio do seriado *The Big Bang Theory*: <https://s3.amazonaws.com/VLIBRAS-WORKLOAD/tbbt.srt>

Legenda - valor máximo:

Filme *Amour*: <https://s3.amazonaws.com/VLIBRAS-WORKLOAD/Amour.srt>

Apêndice C

Tabelas com Resultados dos Experimentos Preliminares

Tabela C.1: Média de tempo de processamento para traduções concorrentes de texto - m1.small

Simulação 1					Simulação 2					Simulação 3					Simulação 4					Simulação 5				
Nº req.	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F				
1	23.59	23.59	0.00	0	24.82	24.82	0.00	0	24.71	24.71	0.00	0	23.60	23.60	0.00	0	24.35	24.35	0.00	0				
2	33.15	34.48	1.32	0	34.74	34.74	0.00	0	40.10	40.12	0.02	0	35.61	35.62	0.01	0	45.27	45.29	0.02	0				
4	65.74	65.78	0.06	0	63.97	64.05	0.10	0	65.80	66.00	0.13	0	65.41	65.53	0.09	0	78.24	78.31	0.05	0				
6	82.85	92.75	13.88	0	83.81	93.96	14.68	0	83.12	95.71	13.76	0	88.27	94.46	3.11	0	118.60	136.69	25.36	0				
10	147.48	151.46	11.06	0	128.66	142.27	19.02	0	156.07	175.99	26.66	0	135.16	143.33	16.01	0	136.06	152.59	21.26	0				
15	178.86	203.04	33.32	0	170.49	186.10	16.34	0	191.65	204.59	12.10	0	175.13	189.80	19.42	0	208.74	224.02	21.59	0				
20	201.96	232.57	24.59	0	187.63	215.93	26.15	0	197.14	226.63	24.53	0	197.39	224.43	35.24	0	204.50	231.57	26.48	0				
25	240.61	281.56	44.32	3	246.32	281.24	32.64	5	264.31	307.04	44.69	4	237.73	274.07	37.88	3	271.46	322.15	61.25	4				
30	275.48	374.34	89.31	0	269.39	367.59	93.86	0	281.33	385.12	93.05	0	277.90	370.44	88.38	0	266.58	361.23	85.48	0				
40	379.93	559.50	153.84	0	334.71	488.62	129.71	0	378.32	578.56	153.36	0	364.50	533.68	144.09	0	318.68	473.11	128.88	0				
50	507.35	797.06	220.20	0	497.98	732.74	202.87	0	491.81	748.29	213.95	0	502.38	744.69	216.71	0	513.62	772.87	213.55	0				

Simulação 6					Simulação 7					Simulação 8					Simulação 9					Simulação 10				
1	23.73	23.73	0.00	0	24.29	24.29	0.00	0	24.87	24.87	0.00	0	23.85	23.85	0.00	0	24.41	24.41	0.00	0				
2	35.20	35.26	0.05	0	36.46	36.52	0.06	0	40.05	40.10	0.05	0	35.53	35.56	0.03	0	34.95	35.00	0.06	0				
4	63.90	64.21	0.36	0	66.39	66.43	0.05	0	75.30	75.34	0.03	0	66.95	67.04	0.05	0	66.04	66.10	0.07	0				
6	83.05	93.22	10.50	0	83.40	94.51	11.97	0	91.03	98.83	10.78	0	89.05	94.40	11.72	0	83.85	95.21	11.80	0				
10	135.83	145.72	20.06	0	130.45	151.29	23.21	0	160.22	182.62	31.27	0	136.51	148.49	16.46	0	129.31	146.11	21.23	0				
15	173.68	187.59	13.80	0	212.55	213.39	1.53	0	215.54	225.90	17.96	0	190.39	202.42	12.20	0	206.91	222.78	21.76	0				
20	192.46	218.88	31.20	0	193.72	225.32	31.90	0	226.77	264.62	49.18	0	212.83	240.40	35.09	0	191.51	225.31	33.91	0				
25	244.42	279.25	40.70	4	244.05	282.09	43.26	4	228.85	273.71	41.14	4	248.09	285.88	37.75	3	277.59	310.74	37.22	3				
30	236.97	329.75	83.00	0	306.54	415.98	103.57	0	269.31	370.51	81.98	0	306.31	393.45	90.18	0	232.24	318.30	71.46	0				
40	294.42	437.58	119.60	0	283.76	421.94	108.71	0	333.51	494.98	137.87	0	329.94	484.95	126.01	0	309.58	466.53	123.02	0				
50	493.58	717.34	184.80	1	537.00	797.32	218.79	0	516.68	771.48	213.36	0	602.09	896.04	251.18	0	529.42	787.91	219.82	0				

Tabela C.2: Média de tempo de processamento para traduções concorrentes de legenda - m1.small

Simulação 1				Simulação 2				Simulação 3				Simulação 4				Simulação 5				
Nº req.	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F
1	175.09	175.09	0.00	0	71.93	71.93	0.00	0	69.59	69.59	0.00	0	67.00	67.00	0.00	0	59.80	59.80	0.00	0
2	330.52	330.52	0.00	0	286.13	286.14	0.00	0	279.68	279.69	0.00	0	278.43	278.43	0.00	0	297.90	297.90	0.00	0
4	591.31	591.31	0.00	0	546.81	546.81	0.00	0	535.47	540.49	8.67	0	508.09	516.25	14.13	0	603.63	603.64	0.00	0
6	1.012.65	1.022.44	19.18	0	944.23	948.88	10.22	0	791.51	798.85	5.16	0	682.08	687.38	10.53	1	792.50	799.62	15.89	0
10	1.335.68	1.335.87	0.31	0	1.125.98	1.126.11	0.14	0	1.166.16	1.166.45	0.36	0	1.017.52	1.018.82	0.64	0	1.180.01	1.180.16	0.22	0
15	1.432.15	1.648.91	318.80	0	1.297.74	1.304.78	9.34	0	1.326.58	1.453.83	159.25	0	1.370.39	1.515.43	207.91	1	1.241.64	1.394.94	175.77	1
20	1.828.38	1.896.72	71.27	3	1.414.71	1.464.88	42.71	3	1.248.03	1.306.36	57.08	3	1.407.73	1.610.88	290.11	1	1.209.54	1.294.82	85.47	2
25	1.795.09	1.990.62	208.59	5	1.699.18	1.735.91	42.66	7	1.416.66	1.502.40	132.36	8	1.945.21	2.148.23	152.48	6	1.657.32	1.806.03	108.72	6
30	2.767.56	3.188.89	343.14	7	2.441.05	2.839.37	356.11	7	1.892.85	2.323.92	343.43	6	2.923.42	3.534.35	484.24	6	2.430.37	2.795.64	298.41	8
40	2.884.34	4.364.85	1.597.40	8	2.849.27	3.909.48	1.481.66	16	2.176.09	3.361.33	1.198.76	4	3.083.35	4.586.07	1.598.30	10	2.733.46	3.946.13	1.477.98	11
50	1.036.79	2.216.86	643.14	41	3.958.09	5.482.69	2.201.51	24	3.813.50	5.573.26	2.318.27	22	7.690.13	8.879.33	2.752.49	29	4.753.69	6.382.16	2.556.48	25
Simulação 6				Simulação 7				Simulação 8				Simulação 9				Simulação 10				
1	72.50	72.50	0.00	0	73.14	73.14	0.00	0	94.89	94.89	0.00	0	66.33	66.33	0.00	0	81.75	81.75	0.00	0
2	296.38	296.38	0.01	0	275.61	275.62	0.00	0	277.59	277.60	0.00	0	157.43	157.43	0.00	1	275.19	275.29	0.10	0
4	542.63	543.81	1.58	0	537.08	542.18	2.95	0	574.48	580.99	11.15	0	424.06	424.07	0.01	1	549.20	554.51	9.19	0
6	1.160.78	1.175.63	14.14	0	809.88	824.28	15.07	0	816.75	817.36	0.41	0	653.97	660.99	14.03	1	814.20	818.14	5.63	0
10	1.516.10	1.516.45	0.43	0	1.257.61	1.257.95	0.39	0	1.334.19	1.334.64	0.57	0	1.183.39	1.183.73	0.85	1	1.321.86	1.322.20	0.48	0
15	1.251.85	1.385.59	290.98	1	1.510.07	1.668.19	268.25	0	1.157.12	1.368.35	286.81	0	1.224.09	1.350.75	210.35	0	1.414.51	1.597.79	197.07	0
20	1.309.44	1.452.81	195.34	2	1.489.41	1.585.59	142.33	2	1.810.86	1.929.89	210.33	2	1.219.19	1.318.26	132.46	3	1.646.83	1.796.03	214.02	1
25	1.219.26	1.317.22	103.99	7	1.824.99	1.930.12	128.30	7	1.663.39	1.832.70	271.04	6	1.413.06	1.480.94	122.38	9	1.566.50	1.763.10	337.88	5
30	2.010.04	2.401.10	360.09	6	2.665.86	3.178.96	617.26	8	2.573.12	3.301.17	856.12	5	1.936.22	2.256.95	259.54	10	2.150.90	2.732.51	599.44	4
40	1.908.98	3.018.79	1.017.44	8	4.829.02	6.306.89	2.132.59	15	3.151.32	4.953.23	1.748.08	6	2.248.67	3.402.66	1.105.13	4	3.971.19	5.406.66	1.937.94	14
50	4.855.54	5.744.51	1.771.00	29	1.089.98	2.237.12	544.40	42	6.475.84	8.832.79	3.416.69	25	3.909.67	5.156.43	1.982.35	28	4.674.99	6.694.70	2.715.98	21

Tabela C.3: Média de tempo de processamento para traduções concorrentes de texto - c3.xlarge

Nº req.	Simulação 1				Simulação 2				Simulação 3				Simulação 4				Simulação 5			
	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F
1	6.31	6.31	0.00	0	4.32	4.32	0.00	0	4.55	4.55	0.00	0	4.58	4.58	0.00	0	4.55	4.55	0.00	0
2	5.71	6.22	0.51	0	3.92	4.03	0.11	0	4.44	4.64	0.20	0	5.51	6.21	0.70	0	4.90	5.13	0.23	0
4	11.31	13.34	2.01	0	6.42	7.32	0.90	0	6.90	7.96	1.06	0	10.30	13.53	3.22	0	6.88	8.42	1.53	0
6	13.95	18.45	4.40	0	8.81	10.78	1.91	0	9.79	11.35	1.51	0	15.79	18.32	2.49	0	9.02	11.34	2.25	0
10	21.09	26.29	5.13	0	14.02	17.83	3.76	0	15.99	18.78	2.72	0	21.09	26.27	5.10	0	14.43	18.29	3.80	0
15	34.40	39.91	5.08	0	22.60	24.81	2.17	0	24.54	27.23	2.33	0	36.32	40.00	3.40	0	23.19	27.62	4.52	0
20	52.40	56.28	6.83	0	32.49	33.63	2.26	0	34.30	36.86	3.49	0	52.23	55.72	7.41	0	41.79	42.44	1.16	0
25	53.46	69.68	12.56	0	31.82	40.01	6.58	0	38.08	45.46	7.07	0	52.72	69.22	12.84	0	34.28	44.02	9.30	0
30	67.51	87.05	19.10	0	39.15	49.37	7.81	0	45.89	55.39	7.92	0	68.91	87.25	17.51	0	44.86	55.91	10.39	0
40	84.51	107.50	21.15	0	53.28	68.34	12.90	0	60.47	73.62	12.55	0	81.42	107.82	23.25	0	56.17	72.04	15.46	0
50	103.05	136.28	27.19	0	65.28	82.03	14.36	0	73.03	89.15	14.07	0	106.33	137.79	29.10	0	67.84	89.51	18.59	0
Nº req.	Simulação 6				Simulação 7				Simulação 8				Simulação 9				Simulação 10			
	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F
1	5.43	5.43	0.00	0	5.32	5.32	0.00	0	4.88	4.88	0.00	0	5.12	5.12	0.00	0	6.87	6.87	0.00	0
2	5.78	6.28	0.50	0	4.90	5.20	0.30	0	5.30	5.41	0.11	0	4.90	5.21	0.31	0	5.55	6.26	0.71	0
4	9.93	11.56	1.62	0	7.57	8.49	0.92	0	7.02	8.13	1.10	0	7.33	8.34	1.00	0	9.38	11.53	2.15	0
6	12.90	15.59	2.69	0	10.84	12.27	1.40	0	10.20	11.70	1.44	0	11.47	13.14	1.64	0	12.67	15.27	2.53	0
10	21.72	29.01	7.19	0	17.17	20.28	2.95	0	15.77	18.79	2.99	0	16.68	19.64	2.89	0	24.10	28.87	4.69	0
15	34.21	41.29	6.63	0	24.50	27.44	2.74	0	25.66	28.09	2.26	0	26.48	26.83	0.25	0	40.84	42.03	2.13	0
20	51.72	53.21	3.59	0	33.72	35.10	3.05	0	34.54	35.93	2.53	0	32.28	34.54	2.92	0	50.47	53.24	5.03	0
25	52.52	67.32	14.54	0	39.64	48.59	7.64	0	39.17	47.97	7.04	0	42.77	50.60	6.41	0	53.48	67.63	12.54	0
30	72.55	87.65	14.17	0	48.64	57.84	8.85	0	51.08	61.20	9.26	0	49.44	55.19	6.25	0	68.71	87.78	18.16	0
40	97.62	117.97	19.28	0	70.58	84.60	12.23	0	65.73	78.07	11.50	0	61.60	74.31	12.22	0	96.99	117.82	18.99	0
50	113.17	139.42	23.90	0	82.82	97.93	14.22	0	80.71	97.76	13.68	0	81.84	99.20	15.87	0	113.10	139.40	24.29	0

Tabela C.4: Média de tempo de processamento para traduções concorrentes de legenda - c3.xlarge

Nº req.	Simulação 1				Simulação 2				Simulação 3				Simulação 4				Simulação 5			
	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F
1	24.28	24.28	0.00	0	25.80	25.80	0.00	0	22.18	22.18	0.00	0	25.70	25.70	0.00	0	22.17	22.17	0.00	0
2	28.18	34.28	6.10	0	24.18	30.69	6.51	1	19.15	20.84	1.70	0	27.27	33.49	6.22	0	25.25	31.56	6.32	0
4	59.58	73.32	13.73	0	53.43	67.71	14.26	0	35.26	38.29	3.01	0	53.14	67.28	14.13	2	46.90	61.08	14.10	0
6	80.54	87.46	6.84	1	84.22	90.76	6.46	0	48.96	49.88	1.13	0	76.45	83.21	6.72	0	72.82	87.40	14.51	0
10	128.28	166.64	38.29	1	124.08	163.02	38.88	1	82.99	87.27	4.24	1	129.63	164.32	34.63	0	114.88	150.46	35.49	0
15	192.46	251.24	62.86	1	192.82	247.00	57.84	2	125.65	146.25	21.96	0	188.28	239.13	54.26	1	172.08	222.81	53.14	0
20	266.02	336.34	70.45	0	260.45	327.86	69.28	0	171.02	180.94	9.82	0	260.52	324.70	64.55	0	251.08	310.06	58.35	0
25	194.55	212.56	17.91	1	267.22	305.89	40.14	2	219.26	226.01	6.62	1	362.64	383.77	21.93	0	268.24	297.76	30.61	0
30	244.07	266.30	22.13	0	512.56	513.59	1.32	0	274.70	277.39	2.32	1	377.69	393.78	15.75	0	338.06	365.37	27.25	0
40	369.25	409.36	39.69	3	419.58	478.01	58.35	1	360.32	373.97	13.59	4	410.53	457.85	47.26	0	392.21	448.33	56.01	0
50	483.29	520.46	37.08	0	471.55	513.61	41.96	0	454.33	464.80	10.36	0	496.34	534.53	38.12	0	450.21	491.74	41.43	0
Nº req.	Simulação 6				Simulação 7				Simulação 8				Simulação 9				Simulação 10			
	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F	TM	PT	DP	F
1	24.75	24.75	0.00	0	24.25	24.25	0.00	0	15.68	15.68	0.00	0	23.36	23.36	0.00	0	22.09	22.09	0.00	0
2	17.74	18.65	0.90	0	18.07	18.87	0.80	0	18.17	25.48	7.31	0	18.97	20.27	1.30	0	21.15	27.66	6.51	0
4	33.29	37.28	4.23	1	35.67	39.88	4.20	0	33.06	45.88	12.83	0	34.09	38.08	4.19	0	44.76	59.06	14.28	0
6	50.82	53.20	2.33	0	50.86	51.49	0.54	0	60.60	72.52	11.90	2	50.07	55.06	4.88	0	88.81	89.51	0.67	0
10	82.99	91.81	8.75	1	84.80	89.39	3.72	1	83.93	117.56	37.50	1	82.49	84.78	2.24	0	99.21	135.32	36.06	0
15	123.64	142.77	20.37	2	127.35	156.85	31.46	0	135.14	183.36	52.15	1	124.97	145.01	20.19	0	148.26	204.77	60.45	0
20	165.75	173.88	7.28	0	168.25	179.60	11.25	0	176.52	205.00	27.93	0	175.61	181.71	5.96	0	226.58	275.47	48.93	0
25	221.46	236.26	15.30	2	226.72	239.39	13.08	0	193.23	217.54	25.19	3	226.87	232.20	5.74	0	228.11	250.40	22.52	0
30	266.54	270.55	3.42	1	271.36	281.11	9.68	0	229.50	251.30	21.74	0	276.53	280.28	3.68	0	277.31	293.53	16.19	0
40	346.87	356.46	9.35	1	363.92	381.74	17.73	2	301.05	340.65	40.48	1	356.62	362.24	5.67	0	366.60	406.80	39.65	1
50	444.36	456.48	12.02	0	449.54	469.01	19.36	0	364.50	404.13	39.53	2	453.28	457.00	3.63	0	426.21	458.01	31.13	1

Apêndice D

Projeto de Experimentos

Tabela D.1: Projeto de Experimentos

Exp	I	A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD	Média	Replicações
1	1	-1	-1	-1	-1	1	1	1	1	1	1	-1	-1	-1	-1	1	Médias das replicações	20 replicações
2	1	-1	-1	-1	1	1	1	-1	1	-1	-1	-1	1	1	1	-1	Média das replicações	20 replicações
3	1	-1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	1	-1	Média das replicações	20 replicações
4	1	-1	-1	1	1	1	-1	-1	-1	-1	1	1	1	-1	-1	1	Média das replicações	20 replicações
5	1	-1	1	-1	-1	-1	1	1	-1	-1	1	1	1	-1	1	-1	Média das replicações	20 replicações
6	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	Média das replicações	20 replicações
7	1	-1	1	1	-1	-1	-1	1	1	-1	-1	-1	1	1	-1	1	Média das replicações	20 replicações
8	1	-1	1	1	1	-1	-1	-1	1	1	1	-1	-1	-1	1	-1	Média das replicações	20 replicações
9	1	1	-1	-1	-1	-1	-1	-1	1	1	1	1	1	1	-1	-1	Média das replicações	20 replicações
10	1	1	-1	-1	1	-1	-1	1	1	-1	-1	1	-1	-1	1	1	Média das replicações	20 replicações
11	1	1	-1	1	-1	-1	1	-1	-1	1	-1	-1	1	-1	1	1	Média das replicações	20 replicações
12	1	1	-1	1	1	-1	1	1	-1	-1	1	-1	-1	1	-1	-1	Média das replicações	20 replicações
13	1	1	1	-1	-1	1	-1	-1	-1	-1	1	-1	-1	1	1	1	Média das replicações	20 replicações
14	1	1	1	-1	1	1	-1	1	-1	1	-1	-1	1	-1	-1	-1	Média das replicações	20 replicações
15	1	1	1	1	-1	1	1	-1	1	-1	-1	1	-1	-1	-1	-1	Média das replicações	20 replicações
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Média das replicações	20 replicações

Tabela D.2: A = -1, B = -1, C = -1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	402.10	699899	104295	3266	898.9
2	246.42	435517	57315	166573.8	26904.2
3	198.30	344068	52538	86189.6	12258.3
4	183.75	319871	47629	109459.4	17574.2
5	262.54	482324	42746	2181	27505.7
6	175.19	309228	41149	102538.2	16134.1
7	165.51	295716	35312	83443.4	2874.8
8	166.15	294184	38122	107159	14249
9	251.43	449195	53657	2614	25449.3
10	201.32	363167	39476	148125	14578.6
11	160.13	275808	44461	80717.4	12945
12	169.98	295949	44008	104984.1	14166.9
13	171.89	293440	50339	107010.4	18567.1
14	169.56	303439	35679	102616.2	10739.4
15	139.41	243125	35695	37000.7	7976.8
16	237.72	432941	42501	4892.2	23869.9
17	138.48	240279	36679	52093.6	3606.2
18	144.60	243558	45650	67944.7	13335.9
19	306.58	552674	60489	694429.5	4120.1
20	155.05	272567	37536	93897.5	1096
Falhas - VLIBRAS					
1					

Tabela D.3: A = 1, B = -1, C = -1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	313.7355	589873	37598	584923.7	17938.6
2	446.5575	850162	42953	544142	17447.3
3	330.659	624313	37005	666972.9	12544.8
4	492.4015	614027	370776	691423.7	756669.3
5	290.2105	529652	50769	706537.2	16323
6	154.996	262770	47222	88243.5	14298.2
7	310.941	560182	61700	690790.9	6915.4
8	549.3685	729429	369308	596456.7	755082.2
9	333.194	282278	384110	111757.4	742354.9
10	169.611	302350	36872	107698	3316.6
11	458.4835	545512	371455	699390.8	756242.9
12	331.1445	290472	371817	110431.9	754207.3
13	513.2285	311290	715167	125137.3	1512057
14	306.722	240535	372909	11820.7	749399.2
15	312.5625	589227	35898	683567.1	2634.3
16	144.71	244355	45065	74270.4	14588.7
17	458.205	886553	29857	1010674.5	2042.1
18	323.8675	274416	373319	42142.2	748924.3
19	325.531	270377	380685	98604.4	748033.9
20	304.557	571516	37598	687724	13239.3
Falhas - VLIBRAS					
21					

Tabela D.4: A = -1, B = 1, C = -1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	926.196	1281813	570579	228396.6	156192.7
2	1053.8635	1493174	614553	301393.2	123668.9
3	904.9245	1301215	508634	217676.5	83155.5
4	849.1945	1152134	546255	346247.2	169117
5	952.461	1369807	535115	181169.4	116535.7
6	918.9495	1226500	611399	404284	215299.2
7	910.9145	1297968	523861	283794.8	83963.2
8	940.987	1409809	472165	235664.9	103288.5
9	868.427	1265832	471022	261448.5	83603.4
10	1126.88	1587119	666641	294432.2	159146.9
11	968.72	1393733	543707	332862.1	60699.2
12	709.7885	917677	501900	362084.3	123139.2
13	1026.965	1493138	560792	252571.6	102603
14	1076.251	1527550	624952	280464	103514.3
15	978.454	1295437	661471	261845.5	223216.4
16	855.442	1215083	495801	379131.4	83779.2
17	825.147	1238750	411544	266557.2	67098.3
18	922.022	1353428	490616	220748.5	60441.2
19	966.586	1432642	500530	296468.9	86145.7
20	986.475	1460647	512303	234965.8	82675.5
Falhas - VLIBRAS					
17					

Tabela D.5: A = -1, B = -1, C = 1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	1,302.26	1998392	606126	3514.7	2737.4
2	687.77	1060962	314576	181024.5	81847.9
3	682.82	907214	458429	285129.3	25396.5
4	690.85	1096263	285435	433371.5	112128.5
5	503.66	844055	163259	218062.6	25472.8
6	382.91	584526	181296	29073.4	3109.5
7	413.69	689679	137704	235310.2	37260.3
8	675.84	918124	433552	56260.8	307580.5
9	719.87	1280835	158914	2986.5	91025.4
10	508.13	821026	195239	295707.5	69099.7
11	432.32	677238	187395	176074.3	48084.6
12	358.34	520175	196509	34308.6	70647.4
13	339.80	512696	166898	105773.8	20830.9
14	367.38	602954	131799	238691.6	48499.2
15	390.92	643559	138272	223242.4	856.3
16	309.27	505065	113474	19617	9103.9
17	328.43	483634	173224	69802.1	37501.3
18	308.70	490264	127127	55699.2	28962.4
19	361.45	531247	191657	62983.2	42744.2
20	457.63	643276	271991	178604.4	25727.8
Falhas - VLIBRAS					
0					

Tabela D.6: A = -1, B = -1, C = -1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	61.29	101043	21533	68502.4	37.7
2	40.05	66778	13312	7374.3	2784.2
3	38.11	64840	11388	3303.1	1067.7
4	36.60	62480	10728	9672.9	1288.8
5	264.67	518647	10691	259981.6	1293.5
6	146.95	281966	11933	496152	129.6
7	201.52	392800	10249	503460	812.3
8	206.87	402521	11211	752513.7	1094.1
9	149.53	287677	11376	482815.6	931.5
10	36.82	62310	11339	4923.6	1045.7
11	38.07	64364	11766	8300.9	803.4
12	35.40	59463	11331	6776.3	1021.1
13	146.45	281071	11823	500073.8	1644.6
14	90.93	170912	10947	253106.9	1018.1
15	150.73	289670	11783	308044.4	754.7
16	36.82	61688	11946	9861.9	827.1
17	146.12	280472	11772	300462.1	891.1
18	32.96	55366	10561	7329.1	937
19	148.85	286645	11054	508756.9	1242.1
20	33.97	56532	11415	4313.8	1010.2
Falhas - VLIBRAS					
16					

Tabela D.7: A = 1, B = 1, C = -1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	1350.355	1903406	797304	564734.8	596279.3
2	1197.0935	1743930	650257	808820.7	576536.2
3	1217.938	1682775	753101	378019.6	690229.3
4	1042.723	1449939	635507	565571.7	622677
5	1093.959	1570022	617896	489291.9	601593
6	1105.1615	1584485	625838	444572.5	608647.4
7	896.7055	1235646	557765	665033.7	427355.9
8	1148.8155	1596656	700975	520997.6	636637.6
9	1199.5145	1640113	758916	469604	658392.1
10	1206.8885	1688299	725478	581175.7	612729.7
11	1372.4775	2049161	695794	956089.4	640308.8
12	1194.4215	1741497	647346	731229.6	490569.3
13	1119.9615	1528543	711380	662226.4	644217.8
14	1055.423	1435324	675522	500488.6	574151.9
15	1114.746	1514460	715032	448890.8	606091
16	1218.88	1721681	716079	357182.8	762234.2
17	1157.9525	1625855	690050	552002.3	626845
18	1123.7045	1674311	573098	439875.3	384741.9
19	1000.906	1582007	419805	573812.7	556911.2
20	1074.4755	1465987	682964	743609.5	465923.5
Falhas - VLIBRAS					
1177					

Tabela D.8: A = -1, B = 1, C = 1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	1432.8115	1948761	916862	889547.3	461707.4
2	1228.7005	1673928	783473	749343.8	405014.3
3	1125.4665	1463609	787324	658779.5	393601.6
4	1379.1965	1854647	903746	837663.9	502090.9
5	1197.6605	1567383	827938	728374.5	407928.3
6	1479.6785	2000984	958373	1004943.4	493293.9
7	1070.882	1483930	657834	572384.7	330723.3
8	1186.2615	1630289	742234	645384.4	321373.3
9	996.6145	1394857	598372	468374.8	342313.9
10	923.069	1283749	562389	548392.3	279089
11	1313.1785	1789984	836373	648392.3	401748.7
12	1670.1895	2309485	1030894	926238.8	501423.8
13	1718.7465	2894747	542746	1435872.2	247973.4
14	1066.641	1483935	649347	793746.7	324598.5
15	1296.6565	1754840	838473	773649.3	419161.5
16	1140.1515	1537430	742873	638262.8	371361.5
17	1309.7345	1743839	875630	732632.9	437740
18	1016.5145	1374636	658393	623782.9	329121.5
19	953.905	1223872	683938	548438.9	341894
20	1578.925	2184227	973623	1008230.3	486736.5
Falhas - VLIBRAS					
32					

Tabela D.9: A = -1, B = -1, C = 1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	88.449	132196	44702	9817.9	8853.2
2	179.416	329493	29339	289900.3	5668.3
3	138.525	243062	33988	240904	8944.1
4	93.628	160303	26953	37476.7	11712.2
5	84.437	134118	34756	29074.2	14753.4
6	136.184	230963	41405	245248.4	16797.3
7	184.408	338199	30617	301014.1	7866.9
8	133.184	233047	33321	234153.5	6417.5
9	85.7785	124721	46836	7241	9224
10	86.3925	121472	51313	15397.3	14571.1
11	88.1555	151821	24490	21133.4	5602.4
12	136.5295	238138	34921	244571.3	10474.5
13	171.4275	287713	55142	356419.1	47231.3
14	136.9945	237954	36035	246213.2	6357.3
15	79.8655	126669	33062	33368.5	10951.1
16	131.4585	231847	31070	255741.8	7440.3
17	96.431	140291	52571	20136.5	57582.1
18	105.56	170772	40348	21317.7	20905.5
19	95.002	165501	24503	18921.4	3851.3
20	129.2275	231367	27088	260814.4	8686.3
Falhas - VLIBRAS					
12					

Tabela D.10: A = 1, B = -1, C = 1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	441.8195	784893	98746	89382	64735.98
2	817.005	1089871	544139	645233.2	708213.2
3	557.8975	997732	118063	660909.7	12170.3
4	761.9425	1053463	470422	620227.2	732788.7
5	412.5695	643917	181222	25412.9	355
6	712.676	1290036	135316	1441861.1	12177.7
7	584.841	1036003	133679	623137.4	63288.3
8	371.174	607139	135209	50969	13941
9	704.592	610371	798813	62809.6	886636
10	517.1795	899829	134530	693213.7	16373.2
11	730.212	1015386	445038	622277.4	743810.1
12	487.629	863350	111908	9413	45213.4
13	532.0665	930612	133521	678411.2	16898.4
14	857.3765	598454	1116299	30907.4	1487519.5
15	422.55	699082	146018	134597	43655.3
16	499.7805	870906	128655	701741.7	16803.8
17	547.893	632927	462859	81951.2	725742.2
18	515.2525	904662	125843	685423	29341.1
19	573.683	686321	461045	134001.8	723406.2
20	558.6795	1018039	99320	620939.3	24922.1
Falhas - VLIBRAS					
21					

Tabela D.11: A = 1, B = -1, C = -1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	157.3875	292328	22447	492582.8	254.4
2	424.9935	838960	11027	845552.3	1061.3
3	150.1405	178511	121770	260162.2	248126.3
4	258.755	395076	122434	510323.3	249570.1
5	264.5875	516494	12681	587887.2	845.8
6	34.162	56475	11849	4547	1099.3
7	149.907	177239	122575	242549.7	245986.8
8	104.274	183037	25511	261866.2	29834.1
9	91.4095	61449	121370	6863	246849.1
10	371.721	510536	232906	616719.7	495177.5
11	147.9765	283799	12154	304184.4	1547.3
12	144.9925	279264	10721	498080.1	88.3
13	143.505	275886	11124	497551.5	1104.6
14	197.622	384459	10785	304605.2	1224.8
15	247.941	373837	122045	485754.1	246401.2
16	89.2505	55830	122671	6572.7	249561.6
17	101.386	79702	123070	22972.5	251644.3
18	88.985	56099	121871	9596.9	249071.7
19	259.4055	507578	11233	994403.8	890.3
20	32.4925	54265	10720	5774.6	146.3
Falhas - VLIBRAS					
32					

Tabela D.12: A = -1, B = 1, C = -1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	496.717	719266	274168	167530.1	147881.5
2	506.171	723242	289100	130787.1	127739.9
3	561.7905	813734	309847	172873.8	140343.9
4	553.227	782524	323930	231155.8	167799.2
5	569.9265	833519	306334	168407.3	159259.7
6	812.7715	1025521	600022	429993.4	379730.6
7	581.123	830575	331671	231406.7	178632.3
8	730.722	982704	478740	377858.1	380242.9
9	580.1165	849886	310347	156229	172311.9
10	619.097	875796	362398	234967.6	219905
11	478.823	700943	256703	227812.9	106583.8
12	613.6	903063	324137	220796.6	173390
13	558.758	820461	297055	175239.4	165633.9
14	622.9745	866030	379919	216514.3	209090.9
15	622.7745	917722	327827	238528.3	164964.1
16	535.1755	809451	260900	193215.4	97951.3
17	806.2635	1013498	599029	387675.4	368237.8
18	559.08	799323	318837	16738.5	159383.8
19	472.2985	698738	245859	246938.9	98894.7
20	599.129	832985	365273	256984.7	167489.3
Falhas - VLIBRAS					
149					

Tabela D.13: A = 1, B = 1, C = 1, D = -1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	1900.2015	2770020	1030383	509839.3	622087.6
2	1820.527	2572072	1068982	570492.8	713843.5
3	1746.0955	2472520	1019671	500353.9	662904.5
4	922.6705	1066367	778974	851861	498693.8
5	830.0832	2123042	698177	830083.2	574953.1
6	1125.4665	1463609	787324	958779.5	490361.6
7	1744.5365	2510729	978344	492199.7	621123
8	1661.832	2410736	912928	623715	503298.8
9	1621.683	2318164	925202	513963	515409.5
10	1576.6065	2289226	863987	583930.3	603346.7
11	1560.4535	2381482	739425	854949.4	552270
12	1490.1655	2282321	698010	934415.6	621957
13	1766.9865	2614593	919380	461968	616022.1
14	1446.773	2079547	813999	958674.2	462962.9
15	1750.11	2527354	972866	627078.2	542715.2
16	1768.2945	2714495	822094	684217.3	520574.6
17	1656.2675	2383913	928622	510168.8	590342.8
18	1569.717	2290545	848889	521397.5	552238.6
19	1674.6945	2436464	912925	536517.8	510147.4
20	1537.935	2286839	789031	568877.5	448461.3
Falhas - VLIBRAS	Req. ã submetidas				
1283	180				

Tabela D.14: A = -1, B = 1, C = 1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	924.4565	1444897	404016	281965.9	163649.5
2	1111.256	1705494	517018	236348.4	381652.9
3	940.9885	1561146	394323	194000.2	190661.6
4	969.7335	1487654	456202	329104.7	190975.5
5	992.883	1483265	575833	311404.4	413752.7
6	1029.3395	1409933	388149	247001.5	147782.1
7	1013.6665	1670530	501151	201384	365785.5
8	915.573	1526182	378456	159035.8	174794.2
9	944.318	1452690	440335	294140.3	175108.1
10	1004.1335	1448301	559966	276440	397885.3
11	992.11	1534221	449999	329290.7	163677.4
12	1178.9095	1794818	563001	283673.2	381680.8
13	1045.388	1650470	440306	241325	190689.5
14	1039.5815	1576978	502185	376429.5	191003.4
15	1097.2025	1572589	621816	358729.2	413780.6
16	966.6945	1499257	434132	294326.3	147810
17	1153.494	1759854	547134	248708.8	365813.4
18	1019.9725	1615506	424439	206360.6	174822.1
19	1014.166	1542014	486318	341465.1	175136
20	1071.787	1537625	605949	323764.8	397913.2
Falhas - VLIBRAS					
152					

Tabela D.15: A = 1, B = -1, C = 1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	195.0895	351383	38796	492726.8	4815.1
2	179.8935	324986	34801	305455.9	2005.5
3	129.621	123516	135726	14646	245200.1
4	180.576	229169	131983	233453.1	245270.5
5	238.714	336747	140681	497062.5	251988.8
6	241.7235	456117	27330	326731.9	5193.6
7	186.1285	337234	35023	487306.2	11386
8	461.7325	895538	27927	1448928.1	5767.1
9	122.9745	111165	134784	9117.8	244592.4
10	237.815	341425	134205	312394.4	253275.1
11	185.8555	235111	136600	233958.1	244558.6
12	176.4015	329307	23496	305109.3	4505
13	237.488	334211	140765	494122.5	245511.5
14	354.4195	566828	142011	1017707.5	244316.1
15	242.978	347386	138570	312409.8	247891.6
16	182.7105	338984	26437	499195.2	5870.4
17	243.586	458591	28581	742001.4	5158
18	70.7135	116564	24863	13321.7	5524.1
19	239.645	343741	135549	298742.3	249507.4
20	525.8315	910506	141157	1750919.7	245136.2
Falhas - VLIBRAS					
36					

Tabela D.16: A = 1, B = 1, C = -1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	595.7295	837854	353605	234648.5	246084.1
2	600.133	847832	352434	297722	257037.7
3	597.9995	823806	372193	361968.9	219729.2
4	559.089	787472	330706	208016.8	255949.2
5	546.6935	779923	313464	242935.9	238160.5
6	571.242	828675	313809	289936.4	219375.1
7	560.3915	778867	341916	228950.9	251727.4
8	641.436	835942	446930	366409.6	324689
9	664.7485	980181	349316	384406.2	249921.9
10	586.4115	902909	269914	327205	208890.7
11	719.7145	961839	477590	292746.8	304182.4
12	724.118	971817	476419	355820.3	315136
13	721.9845	947791	496178	420067.2	277827.5
14	683.074	911457	454691	266115.1	314047.5
15	670.6785	903908	437449	301034.2	296258.8
16	695.227	952660	437794	348034.7	277473.4
17	684.3765	902852	465901	287049.2	309825.7
18	765.421	959927	570915	424507.9	382787.3
19	788.7335	1104166	473301	442504.5	308020.2
20	710.3965	1026894	393899	385303.3	266989
Falhas - VLIBRAS					
1262					

Tabela D.17: A = 1, B = 1, C = 1, D = 1

Replicação	Tempo Médio	Leg - média	Tex - média	Leg - d.p.	Tex - d.p.
1	1076.669	1540536	612802	393476.1	458022.8
2	1044.9665	1582005	507928	343662.8	393586
3	1149.333	1649764	648902	421843.9	46892.4
4	303.487	197362	409612	267132.2	239005.1
5	900.699	1459264	342134	378212.8	184891.8
6	1387.769	1946437	829101	665385.8	431876.8
7	2055.451	2837439	1273463	1280823.1	805930.6
8	1149.403	1649937	648869	422016.9	46877.4
9	1563.8235	2027709	1099938	680823.1	705930.6
10	900.769	1459437	342101	378385.8	184876.8
11	1373.3875	1849837	896938	554039.2	485990.9
12	886.037	1362537	409537	267849.2	238938.5
13	974.685	1438552	510818	339091.5	358587.4
14	942.9825	1480021	405944	289278.2	294150.6
15	783.953	1260278	307628	212747.6	139569.7
16	798.715	1357280	240150	323828.2	85456.4
17	1285.785	1844453	727117	611001.2	332441.4
18	1461.8395	1925725	997954	626438.5	606495.2
19	798.785	1357453	240117	324001.2	85441.4
20	1076.739	1540709	612769	393649.1	458007.8
Falhas - VLIBRAS					
1112					

Apêndice E

Configuração do Ambiente de Experimentos

Código Fonte E.1: Arquivo de configuração do Tomcat 7 (/etc/default/tomcat7)

```
1 # anterior
2 # JAVA_OPTS="-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC"
3
4 # atual
5 JAVA_OPTS="-Djava.awt.headless=true -Xmx2048m -XX:+UseConcMarkSweepGC"
```

Código Fonte E.2: Linha de comando do ab para simular o 1º experimento para A(-1)/B(1)/C(-1)/D(-1)

```
1 #Teste: sem injecao de falhas, com 500 requisicoes simultaneas, carga de trabalho baixa,
   instancia Ec2 ml.small
2 ab -c 250 -l -n 250 -p carga-de-trabalho/texto-small.json -q -r -s 10800 -T application/
   json http://184.169.148.166/Broker/requests > resultados/mlsmall/500/sem\ falhas/carga\
   baixa/1/texto.txt & ab -c 250 -l -n 250 -p carga-de-trabalho/legenda-tbbt.json -q -r -
   s 10800 -T application/json http://184.169.148.166/Broker/requests > resultados/mlsmall
   /500/sem\ falhas/carga\ baixa/1/legenda.txt
```

Código Fonte E.3: Arquivo de saída do ab

```
1 This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
2 Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
3 Licensed to The Apache Software Foundation, http://www.apache.org/
4
5 Benchmarking 184.169.148.166 (be patient).....done
6
7
```

Tabela E.1: Parâmetros utilizados no ab

Parâmetro	Significado
-c	Nível de concorrência.
-l	Não reportar erros se o tamanho da página de resposta não for constante. Utilizado para páginas dinâmicas, que se aplica ao nosso caso já que o serviço sempre retorna uma <i>url</i> diferente.
-n	Nº de requisições a ser submetidas.
-p	Arquivo a ser submetido na requisição HTTP POST.
-q	Omitir uma contagem de progresso quando enviar mais de 150 requisições.
-r	Não terminar ao receber mensagens de erros de <i>sockets</i> .
-s	Nº máximo de segundos antes do <i>socket</i> expirar (<i>timeout</i>).
-T	Cabeçalho do tipo de conteúdo (<i>Content-type</i>).

```

8  Server Software:      Apache-Coyote/1.1
9  Server Hostname:     184.169.148.166
10 Server Port:         80
11
12 Document Path:       /Broker/requests
13 Document Length:     Variable
14
15 Concurrency Level:    250
16 Time taken for tests:  2355.075 seconds
17 Complete requests:    250
18 Failed requests:      0
19 Total transferred:    54471 bytes
20 Total body sent:      62000
21 HTML transferred:     22471 bytes
22 Requests per second:  0.11 [#/sec] (mean)
23 Time per request:     2355074.682 [ms] (mean)
24 Time per request:     9420.299 [ms] (mean, across all concurrent requests)
25 Transfer rate:        0.02 [Kbytes/sec] received
26                       0.03 kb/s sent
27                       0.05 kb/s total
28
29 Connection Times (ms)
30      min  mean[+/-sd] median   max
31 Connect:    4   316 461.5      6   1000
32 Processing: 730172 1281497 228436.4 1298199 2355067
33 Waiting:    730172 1281497 228437.2 1298199 2355067
34 Total:      730178 1281813 228396.6 1298309 2355073
35
36 Percentage of the requests served within a certain time (ms)
37  50%    1298309

```

38	66%	1406139
39	75%	1449017
40	80%	1475671
41	90%	1516974
42	95%	1533905
43	98%	1653210
44	99%	1669527
45	100%	2355073 (longest request)
