Desenvolvimento de mecanismos para auxiliar no
estudo e seleção de técnicas de armazenamento
distribuído de arquivos em ambientes de
computação multinuvem

Rafael Mira de Oliveira Libardi

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Rafael Mira de Oliveira Libardi

Desenvolvimento de mecanismos para auxiliar no estudo e seleção de técnicas de armazenamento distribuído de arquivos em ambientes de computação multinuvem

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Julio Cesar Estrella

USP – São Carlos
Novembro de 2015

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

L694d Libardi, Rafael Mira de Oliveira
Desenvolvimento de mecanismos para auxiliar
no estudo e seleção de técnicas de armazenamento
distribuído de arquivos em ambientes de computação
multinuvem / Rafael Mira de Oliveira Libardi;
orientador Julio Cesar Estrella. - São Carlos -
SP, 2015.
109 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2015.

1. Armazenamento de arquivos. 2. Computação
em nuvem. 3. Dispersão de dados. 4. Avaliação de
desempenho. I. Estrella, Julio Cesar, orient. II.
Título.

Rafael Mira de Oliveira Libardi

Development of mechanisms to ease the study and
selection of distributed file storage techniques in muticloud
computing environments

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Julio Cesar Estrella

USP – São Carlos
November 2015

*Este trabalho é dedicado às pessoas que, ao
meu lado, me ajudam a conquistar as coisas
pelas quais luto ao acordar todas as manhãs
Em especial, aos meus pais, familiares, amigos, professores e mentores.
Pessoas incríveis e especiais, com as quais tive o prazer de conviver.*

AGRADECIMENTOS

Este trabalho de mestrado certamente não teria sido possível sem a participação de várias pessoas e amigos conhecidos durante a graduação. Meus sinceros agradecimentos: Aos professores e mentores, com destaque àqueles que, com sua simpatia e métodos didáticos, ensinaram conceitos importantes. Em especial ao meu orientador Julio Cezar Estrella, que tanto me apoiou e me guiou durante este período e ao Stephan Reiff-Marganiec, que me recebeu tão bem em Leicester e me auxiliou durante o estágio de pesquisa desenvolvido no exterior.

Aos amigos, sem os quais este período de minha vida teria sido muito menos proveitoso e divertido. Agradecimentos especiais a Adriana Ansolin, Ariana Leonel, Carol Bianco, Carlota, Danilão, Eduardo Moraes, Edgard Regolão Jr., Évert Bacchini, Felipe Guedes, Lara Santi, Leandro Lourenço, Lucas Cordeiro, Luiza Bohnen, Maksuel Boni, Murilo Ramiro, Natasha Leal, Pimpão, Regina Duarte, Ricardo Pallone, Sérgio Arruda, Thiago Coelli, Thiago Christoff, Victor Martins, entre tantos outros!

Além desses, agradeço também a toda minha família e a Deus, pelo apoio, confiança e oportunidades a mim depositados, sempre. Sem a ajuda de todos vocês, a experiência única que tive nesse período de minha vida não seria tão proveitosa e completa!

Agradeço também ao CNPq, Capes e o apoio da FAPESP através dos processos 2013/05567-9 e 2014/05888-2.

“I do not think there is any thrill that can go through the human heart like that felt by the inventor as he sees some creation of the brain unfolding to success. Such emotions make a man forget food, sleep, friends, love, everything.”
(Nikola Tesla)

RESUMO

LIBARDI, R. M. O.. **Desenvolvimento de mecanismos para auxiliar no estudo e seleção de técnicas de armazenamento distribuído de arquivos em ambientes de computação multinuvem.** 2015. 109 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O armazenamento de arquivos e dados está seguindo um paradigma de mudança para a Internet utilizando a nuvem. Porém, este processo ainda gera algumas dúvidas em relação à segurança e a disponibilidade dos arquivos. Este projeto de mestrado propõe criar uma camada de abstração entre diversos servidores de armazenamento público, sem infraestrutura adicional, e possibilitar ao usuário comum um método de armazenamento distribuído, com maior redundância e segurança se comparada às abordagens atuais de armazenamento. Outro aspecto importante deste projeto foi criar uma ferramenta que seja flexível o suficiente para ser fácil de ser utilizada por outros pesquisadores para avaliar novas técnicas de armazenamento e seleção automática de estratégias de dispersão. Este projeto foi feito utilizando conceitos de dispersão de arquivos e de mecanismos de seleção automática utilizando atributos de QoS. A primeira contribuição foi um protótipo denominado FlexSky que implementa os mecanismos de dispersão utilizados. Devido a quantidade grande de parâmetros para se escolher no processo de dispersão, foi necessário desenvolver um mecanismo que reduzisse a quantidade de parâmetros para que um usuário comum consiga utilizar a ferramenta. Este mecanismo foi a segunda contribuição (MSSF), no qual foi criado um modelo baseado em otimização linear inteira para realizar a seleção automática de estratégias de armazenamento multinuvem. Para se avaliar as contribuições foram propostos testes qualitativos de usabilidade da ferramenta e teste quantitativos de desempenho dos processos da ferramenta. Os resultados sugerem que a ferramenta FlexSky facilitou o estudo destes mecanismos de dispersão e possibilita uma avaliação de desempenho deles. Já o MSSF consegue a partir de atributos de QoS escolher uma estratégia ótima que maximize os parâmetros desejados do processo de dispersão para diversos casos considerando diferentes quantidades de módulos e de provedores.

Palavras-chave: Armazenamento de arquivos, Computação em nuvem, Dispersão de dados, Avaliação de desempenho.

ABSTRACT

LIBARDI, R. M. O.. **Desenvolvimento de mecanismos para auxiliar no estudo e seleção de técnicas de armazenamento distribuído de arquivos em ambientes de computação multinuvel**. 2015. 109 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Data and file storage is moving to the Internet using the cloud approach. However, this process still creates issues related to security and availability. This project creates an abstraction layer among several public storage servers, without the need for additional infrastructure, enabling regular users to use a distributed storage method with increased redundancy and security comparing to current techniques. Another important aspect of this project was to create a tool flexible enough to be easy for other researchers to evaluate new dispersal techniques. The first contribution was a prototype named FlexSky, which implements the studied dispersal mechanisms. Due to the big amount of parameters to choose from during the dispersal process, it was needed to develop a mechanism to reduce the amount of user input parameters so a regular user can use the tool. This mechanism was the second contribution (MSSF) and it is composed of a model based on integer linear optimization that automatically selects multicloud dispersal strategies. The evaluation phase was composed of qualitative usability tests and quantitative performance tests. The results suggests that the FlexSky tool made the study of dispersal mechanisms easier and enables to evaluate their performance. MSSF results shown that using QoS attributes, MSSF can choose an optimal dispersal strategy which maximizes and minimizes the required QoS attributes considering several modules and providers.

Key-words: File storage, Cloud computing, Data dispersal, Performance evaluation.

LISTA DE ILUSTRAÇÕES

Figura 1 – Princípios da segurança da informação (Adaptado de Stallings (2002)) . . .	30
Figura 2 – Arquitetura típica de um ambiente em nuvem (Adaptado de Kumar <i>et al.</i> (2012))	32
Figura 3 – Funcionamento básico de um algoritmo de dispersão de informações (Adaptado de Rabin (1989a), Rabin (1990))	33
Figura 4 – Funcionamento básico de um algoritmo de dispersão de informações (Adaptado de Rabin (1989a), Rabin (1990)). Mesmo com as partes C e F indisponíveis, ainda é possível recuperar o arquivo.	33
Figura 5 – FlexSky - Interface gráfica	58
Figura 6 – FlexSky - Arquitetura geral	58
Figura 7 – FlexSky - Arquitetura do <i>Experiment Planner</i>	59
Figura 8 – FlexSky - Arquitetura interna do Flexible Blob Storage (Parte 1)	61
Figura 9 – Flexible Blob Storage Internal Architecture (Part 2)	63
Figura 10 – Diagrama de Classes UML simplificado mostrando as classes do módulo Blob Storage	65
Figura 11 – Diagrama de Classes UML simplificado mostrando as classes que representam os tipos de dados do sistema	66
Figura 12 – Diagrama de Classes UML simplificado mostrando as classes que gerenciam os metadados	67
Figura 13 – Diagrama de Classes UML simplificado mostrando as classes do módulo Experiment Planner	68
Figura 14 – Diagrama de Classes UML simplificado mostrando as classes que gerenciam e coordenam o sistema	69
Figura 15 – Diagrama de Classes UML simplificado mostrando as classes do módulo MSSF	70
Figura 16 – Diagrama de Classes UML simplificado mostrando as classes do módulo Logger	71
Figura 17 – Diagrama de Casos de Uso UML simplificado mostrando as operações do usuário	72
Figura 18 – Interface de usuário do MSSF	73
Figura 19 – Fluxograma simplificado do sistema especialista baseado em regras	77
Figura 20 – Fluxo de trabalho do MSSF	79
Figura 21 – Fluxograma simplificado da metodologia	80
Figura 22 – Tempo das operações da FlexSky	86

Figura 23 – Vazão das operações na FlexSky	87
Figura 24 – Gráfico de superfície Rabin IDA	87
Figura 25 – Gráfico de superfície Replicate IDA	88
Figura 26 – Análise do módulo Pipe Base64	89
Figura 27 – Vazão dos provedores de nuvem e Sobrecarga de armazenamento	89
Figura 28 – Tempo de seleção	92
Figura 29 – Influência estatística dos fatores	93

LISTA DE TABELAS

Tabela 1 – Classificação dos serviços na nuvem disponíveis. (Adaptado de Gao <i>et al.</i> (2010))	31
Tabela 2 – Técnicas de seleção de serviços em nuvem (Adaptado de Alabool e Mahmood (2013))	40
Tabela 3 – Principais métricas encontradas na literatura de seleção de nuvens (Adaptado de Alabool e Mahmood (2013))	41
Tabela 4 – Influências das categorias de seleção nos parâmetros de QoS	44
Tabela 5 – Resumo dos tipos de arquivos e suas características	45
Tabela 6 – Características dos trabalhos semelhantes	51
Tabela 7 – Variáveis do modelo de otimização	76
Tabela 8 – Provedores	77
Tabela 9 – Perguntas de usabilidade e suas respostas	85
Tabela 10 – Fatores e níveis variáveis utilizados no experimento.	86
Tabela 11 – MSSF - Fatores e níveis utilizados nos experimentos.	91
Tabela 12 – Resultados do teste de usabilidade	94
Tabela 13 – Organização das atividades	98

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Considerações Iniciais	23
1.2	Contextualização	23
1.3	Motivação	24
1.4	Objetivos	26
1.4.1	<i>Modelo de armazenamento de arquivos multinuvem - FlexSky . . .</i>	<i>26</i>
1.4.2	<i>Seleção de estratégias de dispersão multinuvem de arquivos - MSSF</i>	<i>26</i>
1.5	Organização do trabalho	27
1.6	Considerações Finais	27
2	REVISÃO DA LITERATURA	29
2.1	Considerações Iniciais	29
2.2	Modelo de armazenamento de arquivos multinuvem - FlexSky . . .	29
2.2.1	<i>Segurança em ambientes de computação em nuvem</i>	<i>29</i>
2.2.2	<i>Armazenamento de arquivos na nuvem</i>	<i>31</i>
2.2.3	<i>Dispersão de informações</i>	<i>32</i>
2.2.4	<i>Algoritmos e códigos de dispersão de informações</i>	<i>34</i>
2.2.5	<i>Campos Finitos</i>	<i>36</i>
2.2.6	<i>Aplicações da dispersão de arquivos em ambientes distribuídos . . .</i>	<i>37</i>
2.2.7	<i>Avaliação do armazenamento de arquivos na nuvem</i>	<i>38</i>
2.2.8	<i>Métricas de Desempenho e Usabilidade</i>	<i>39</i>
2.3	Seleção de estratégias de dispersão multinuvem de arquivos - MSSF	40
2.3.1	<i>Métodos de seleção e avaliação de nuvens</i>	<i>40</i>
2.3.2	<i>Métricas de seleção de nuvens</i>	<i>41</i>
2.3.2.1	<i>Custos</i>	<i>41</i>
2.3.2.2	<i>Sobrecarga de Armazenamento</i>	<i>42</i>
2.3.2.3	<i>Disponibilidade e durabilidade</i>	<i>42</i>
2.3.2.4	<i>Desempenho</i>	<i>43</i>
2.3.2.5	<i>Confidencialidade</i>	<i>44</i>
2.3.2.6	<i>Classificação de arquivos</i>	<i>44</i>
2.4	Considerações finais	45
3	TRABALHOS RELACIONADOS	47

3.1	Considerações Iniciais	47
3.2	Trabalhos relacionados com a FlexSky	47
3.2.1	<i>High-availability Integrity Layer (HAIL)</i>	47
3.2.2	<i>Redundant Array of Cloud Storage - RACS</i>	48
3.2.3	<i>iDataGuard</i>	48
3.2.4	<i>DepSky</i>	48
3.2.5	<i>Secured Virtual Diffused File System - SVDFS</i>	48
3.2.6	<i>CloudSafe</i>	49
3.2.7	<i>SCFS</i>	49
3.2.8	<i>NubiSave</i>	49
3.2.9	<i>Cloud-RAID</i>	49
3.2.10	<i>Bonfire</i>	50
3.2.11	<i>Diferenciais deste trabalho</i>	50
3.3	Trabalhos relacionados com o MSSF	52
3.3.1	<i>Scalia</i>	52
3.3.2	<i>Diferenciais deste trabalho</i>	53
3.4	Considerações Finais	53
4	DESENVOLVIMENTO DO TRABALHO	55
4.1	Considerações Iniciais	55
4.2	Contribuições do projeto de pesquisa	55
4.3	Modelo de armazenamento de arquivos multinuvel - FlexSky	57
4.3.1	<i>Arquitetura</i>	57
4.3.2	<i>Módulo Experiment Planner</i>	59
4.3.3	<i>Módulo Flexible Blob Storage</i>	61
4.3.4	<i>Módulo Logger</i>	63
4.3.5	<i>Modelagem</i>	64
4.4	Seleção de estratégias de dispersão multinuvel de arquivos - MSSF	73
4.4.1	<i>Parâmetros de entrada do usuário</i>	73
4.4.2	<i>Modelo e motor de otimização</i>	74
4.4.3	<i>Obtenção dos parâmetros dos provedores</i>	76
4.4.4	<i>Sistema especialista de ajuste dos pesos dos parâmetros de QoS baseado em regras</i>	77
4.4.5	<i>Fluxo de trabalho</i>	78
4.5	Metodologia	78
4.6	Ferramentas para o Desenvolvimento	80
4.7	Considerações Finais	81
5	AVALIAÇÃO EXPERIMENTAL	83
5.1	Considerações Iniciais	83

5.2	Avaliação da FlexSky - Estudo de caso	83
5.2.1	<i>Ambiente de testes</i>	83
5.2.2	<i>Metodologia</i>	84
5.2.3	<i>Resultados</i>	86
5.3	Avaliação do <i>Multi cloud Storage Selection Framework (MSSF)</i> . .	90
5.3.1	<i>Ambiente de testes</i>	90
5.3.2	<i>Avaliação de desempenho</i>	91
5.3.3	<i>Teste de Usabilidade</i>	92
5.4	Considerações Finais	94
6	CONCLUSÕES E TRABALHOS FUTUROS	95
6.1	Considerações iniciais	95
6.2	Conclusão	95
6.3	Contribuições	96
6.4	Integração com Outras Pesquisas	96
6.5	Trabalhos futuros	97
6.6	Etapas cumpridas	97
6.6.1	<i>Cronograma</i>	97
6.7	Considerações finais	98
7	PUBLICAÇÕES E SUBMISSÕES	101
	REFERÊNCIAS	103

INTRODUÇÃO

1.1 Considerações Iniciais

Este capítulo contextualiza o trabalho, discutindo os objetivos à serem alcançados juntamente com uma discussão da motivação e da importância que esta pesquisa possui, ressaltando os aspectos cruciais do armazenamento de arquivos. Também é apresentada neste capítulo a organização da dissertação.

1.2 Contextualização

A computação em nuvem surgiu nos últimos anos como um novo paradigma computacional que busca satisfazer a recente promessa de computação por demanda e utilidade. Com o surgimento deste paradigma, cada vez mais organizações estão buscando migrar seus sistemas existentes e seus dados para a nuvem com o objetivo de reduzir o custo com infraestrutura de tecnologia da informação (GENS, 2008; ARMBRUST *et al.*, 2010).

Aliado à esta tendência de terceirizar a gestão de informações, é possível observar um rápido crescimento de diversas empresas que oferecem serviços de computação em nuvem. Dentre elas, pode-se destacar a *Google*, que oferece o serviço *Google Drive* (GOOGLE©, 2013b) para o armazenamento na nuvem e o *Google Docs* (GOOGLE©, 2013a), que oferece uma suíte de aplicativos para escritório também na nuvem.

Como pode ser observado, há uma tendência de migração de serviços e aplicações existentes para a nuvem, como uma tentativa de se obter mais disponibilidade, integridade e confidencialidade da informação. Disponibilidade é uma propriedade que garante que a informação estará sempre acessível para uso legítimo, ou seja, pelos usuários autorizados pelo proprietário da informação (STALLINGS, 2002). Já a integridade é uma propriedade que determina que a informação manipulada contenha todas as características originais estabelecidas pelo proprietário

da informação. Por fim, a confidencialidade é a propriedade que garante que somente as pessoas autorizadas, ou seja, apenas entidades legítimas que tenham permissão expressa pelo proprietário terão acesso a informação armazenada. Estas são as três propriedades que os administradores e gestores da informação mais desejam e priorizam na escolha da melhor maneira e local para armazenar os dados (GENS, 2008).

Porém, na computação em nuvem, ainda não é possível garantir todas estas propriedades (GENS, 2008). Se por um lado há ganho na redução dos custos e na melhora do desempenho, por outro perde-se no controle e na gestão dos dados, que passa do controle interno (modelo tradicional) para o controle externo (modelo em nuvem), o que acaba desmotivando diversas entidades (usuários ou organizações) a mudar o paradigma de armazenamento.

É possível observar que devido à escolha de manter empresas terceiras gerenciando os recursos computacionais e o armazenamento das informações, o risco de roubo e vazamento de informações está cada vez maior e está preocupando organizações que querem garantir que todos os seus dados estejam protegidos contra roubos e acesso de pessoas não autorizadas (GENS, 2008).

Atualmente, não há garantia técnica alguma de que os dados armazenados não serão expostos para pessoas não autorizadas, seja para o profissional de infraestrutura da empresa prestadora do serviço, ou mesmo possíveis atacantes. O que se tem são contratos estabelecidos entre as empresas que devem garantir que os dados não serão acessados por pessoas não autorizadas. Esta sensação de não ter controle dos próprios dados é o maior obstáculo para a adoção em massa da computação em nuvem (GENS, 2008).

O problema de garantir o desempenho e a segurança de sistemas de armazenamento em nuvem ainda não foi tão bem explorado, a ponto de se ter uma solução considerada adequada para a maioria das situações.

Usualmente, soluções que privilegiam a segurança acabam não sendo muito eficientes em determinados contextos, como pode ser visto em Yu *et al.* (2013) e G.Nalinipriya e Kumar (2013), nos quais são propostas abordagens que criptografam os arquivos na nuvem. Tais abordagens, apesar de sua relativa segurança, não são muito eficientes quando se trata de arquivos grandes e chaves grandes. O tempo utilizado para criptografar e descriptografar estes arquivos é considerado elevado e portanto, inviável de ser feito constantemente e em larga escala. Infelizmente, ainda não há uma abordagem adequada para guardar dados que tenha uma boa relação segurança-desempenho para sistemas executados na nuvem.

1.3 Motivação

Neste contexto, surge a necessidade de um modelo de armazenamento seguro de arquivos na nuvem, que seja ao mesmo tempo eficiente e possua um grau de segurança mais elevado em

relação aos métodos atuais.

Uma das abordagens para tentar melhorar o armazenamento de arquivos na nuvem é o armazenamento de forma dispersa dos arquivos. Neste tipo de solução, o arquivo é dividido em partes, e essas, por sua vez, são armazenadas em diferentes servidores. Com isso é possível obter soluções de armazenamento de arquivos em nuvem que sejam redundantes, possuam melhor eficiência e também sejam mais seguras.

Um dos trabalhos desenvolvidos com esta abordagem foi o de [Mar \(2011\)](#), em que se criou um sistema virtual de arquivos seguro e difuso para ambientes em nuvem. Este sistema garante a clara distinção entre os donos dos arquivos e os donos da infraestrutura de armazenamento. Com isso, apenas o dono do arquivo consegue acessá-lo, evitando que pessoas não autorizadas provenientes dos provedores consigam obter acesso ao conteúdo do mesmo.

Porém, apesar de conseguir bons níveis de segurança, confidencialidade e tolerância a falhas há uma limitação importante. Todo o sistema utiliza um servidor de registro em que são concentradas todas as informações necessárias para acessar os arquivos armazenados. Devido a esta escolha, este servidor é o ponto fraco desta arquitetura, pois uma vez que se consegue acesso a ele, se tem acesso a todos os arquivos armazenados na arquitetura. Esta arquitetura não é muito flexível, pois necessita de uma infraestrutura própria (servidor de registro) e não se pode customizar o nível de segurança e disponibilidade desejado.

Criar uma arquitetura de dispersão descentralizada foi a motivação inicial deste trabalho. Durante o desenvolvimento, também foi notado por meio de pesquisa na literatura da área, a ausência de uma ferramenta flexível e fácil o suficiente para se estudar mecanismos de dispersão de arquivos em nuvem. E esta foi a segunda motivação para criar a plataforma FlexSky. A ferramenta é baseada em uma arquitetura totalmente descentralizada e flexível o suficiente para se estudar facilmente novos mecanismos de dispersão no armazenamento em nuvem, como algoritmos de dispersão, compressão e criptografia.

Ainda durante o desenvolvimento do trabalho, outro aspecto explorado foi o problema de seleção de estratégias de dispersão em ambientes multinuem. Com este novo paradigma, o usuário era forçado a tomar diversas decisões técnicas sobre a maneira com que seria feita a dispersão dos arquivos. Isto se tornou um problema, pois afugentava usuários sem conhecimentos técnicos. Assim, durante o estágio de pesquisa no exterior (BEPE) foi proposto estudar e criar um mecanismo de seleção automático de uma estratégia de dispersão que fosse adequado para usuários leigos. Esta foi a motivação para se criar o MSSF (*Multi-cloud Storage Selection Framework*), um seletor baseado em programação linear inteira que seleciona automaticamente a melhor estratégia de dispersão. Este seletor faz parte da ferramenta FlexSky para ser testado e avaliado.

1.4 Objetivos

Este trabalho apresenta dois objetivos principais. O primeiro está relacionado à criação de um modelo de armazenamento de arquivos na nuvem e o segundo com a criação de um mecanismo de seleção de estratégias de dispersão.

1.4.1 *Modelo de armazenamento de arquivos multinuvem - FlexSky*

O primeiro objetivo deste trabalho foi propor, modelar, desenvolver e avaliar um novo modelo de armazenamento de arquivos na nuvem. Este modelo é baseado na difusão de arquivos entre diversos provedores de serviços, de maneira a se criar uma nuvem virtual, que possua níveis melhores de segurança, privacidade, desempenho e disponibilidade. Porém, ao contrário de outros modelos do tipo, este será fundamentado no uso de uma composição de chave única que armazenará todos os metadados necessários para se acessar o arquivo, sem a necessidade de nenhuma infraestrutura adicional e possibilita flexibilizar os níveis de segurança, disponibilidade e privacidade para cada usuário. Com este modelo, é possível garantir que apenas o usuário com acesso aos meta-dados conseguirá ter acesso aos arquivos, evitando o acesso indevido.

Outra característica do modelo é que ele seja flexível o suficiente para facilitar o estudo de novos algoritmos de dispersão, módulos de compressão, criptografia, módulos de transformações nos arquivos e estratégias de seleção no ambiente de armazenamento multinuvem. Este modelo gerou um protótipo que será utilizado pela comunidade científica para facilitar o estudo de novos mecanismos de armazenamento multinuvem.

Esta parte do trabalho foi dividida em quatro fases distintas, porém complementares. Primeiramente foi proposto um modelo distribuído de armazenamento que contenha as características desejadas (segurança, disponibilidade, desempenho e flexibilidade). Posteriormente, este modelo foi formalizado com o intuito de se obter modelos conceituais, diagramas *UML* e diagramas de arquitetura. Após esta fase e uma extensa análise da viabilidade do modelo proposto, ele foi implementado e posteriormente avaliado.

1.4.2 *Seleção de estratégias de dispersão multinuvem de arquivos - MSSF*

O segundo objetivo do trabalho foi o desenvolvimento de um mecanismo de seleção de estratégia de armazenamento multinuvem que levasse em consideração aspectos únicos deste tipo de ambiente e reduzisse a necessidade de inserção de parâmetros provenientes do usuário, facilitando o uso desta nova técnica de armazenamento. Este mecanismo deve apresentar desempenho que seja compatível com ambientes multinuvem, sendo possível sua utilização em ambientes de grande escala, sem perda de desempenho elevada.

Esta parte do trabalho também utilizou quatro etapas. Primeiro houve um estudo da

literatura de seleção de serviços com o intuito de se obter o arcabouço necessário para propor um mecanismo de seleção adequado. Posteriormente, este mecanismo foi formalizado gerando um modelo matemático de otimização. Este modelo foi então desenvolvido e aplicado na arquitetura, sendo possível gerar resultados de uma avaliação qualitativa e quantitativa.

1.5 Organização do trabalho

Esta monografia está dividida em seis capítulos, sendo que cada capítulo é dividido em duas seções, cada uma com o conteúdo relevante para se obter as duas contribuições desejadas (FlexSky e MSSF). O capítulo 1 apresenta a contextualização motivação e objetivos do projeto de mestrado. No capítulo 2 é apresentada uma revisão da literatura buscando criar um arcabouço inicial de conceitos para o melhor entendimento do projeto de mestrado. Trabalhos relacionados com o projeto são comentados e analisados no capítulo 3, juntamente com uma análise comparativa mostrando o diferencial do presente trabalho em relação aos demais encontrados na literatura. O desenvolvimento do trabalho, incluindo algumas definições do sistema é apresentado no capítulo 4, juntamente com sua modelagem e desenvolvimento. Uma avaliação experimental é apresentada no capítulo 5 mostrando a viabilidade do protótipo inicial e comparando algumas abordagens. Por fim, no capítulo 6 são apresentadas possíveis desdobramentos futuros juntamente com o cronograma de desenvolvimento e integrações com outras pesquisas do laboratório. No capítulo 7 são apresentadas as publicações e artigos prontos para submissão oriundos deste projeto de mestrado.

1.6 Considerações Finais

Neste capítulo foram abordados os conceitos introdutórios do projeto de mestrado e a importância desta pesquisa na sociedade moderna que está mudando o paradigma de armazenamento de arquivos em ambientes computadorizados. Ainda, este capítulo abordou os objetivos da pesquisa e todo o conhecimento que será gerado para a literatura da área.

REVISÃO DA LITERATURA

2.1 Considerações Iniciais

Este capítulo contém uma revisão da literatura necessária para o entendimento deste projeto de mestrado. A proposta desta seção é construir um arcabouço inicial de conceitos e técnicas para compreender e possibilitar uma análise crítica consistente do projeto.

2.2 Modelo de armazenamento de arquivos multinuem - FlexSky

Esta seção apresenta o arcabouço necessário para compreender o modelo de dispersão de arquivos descentralizado desenvolvido durante o projeto.

2.2.1 *Segurança em ambientes de computação em nuvem*

De acordo com Stallings ([STALLINGS, 2002](#)), a segurança em ambientes computacionais pode ser dividida em alguns pontos principais baseados na arquitetura de segurança OSI ([WARD, 1989](#)). Estes pontos são:

ATAQUES À SEGURANÇA: Estes ataques podem ser divididos em dois tipos principais: ataques passivos, como a leitura não autorizada de mensagens ou arquivos e análise de tráfego; e ataques ativos, como modificação de mensagens ou arquivos e ataques de negação de serviço;

MECANISMO DE SEGURANÇA: Qualquer processo ou dispositivo capaz de detectar, impedir ou permitir a recuperação de um ataque à segurança;

SERVIÇOS DE SEGURANÇA: Incluem autenticação/controle de acesso, confidencialidade de dados, integridade de dados, irretratabilidade e disponibilidade;

Figura 1 – Princípios da segurança da informação (Adaptado de Stallings (2002))

Os serviços de segurança possuem algumas propriedades básicas, que devem ser garantidas para uma completa e eficaz implementação de segurança em ambientes computacionais. A Figura 1 apresenta estas propriedades que são descritas detalhadamente a seguir:

CONFIDENCIALIDADE: Trata-se da ocultação de informações ou de recursos, protegendo-os de acesso não autorizado, como por exemplo, criptografar um arquivo, evitando que ele seja acessado por pessoas não autorizadas;

AUTENTICIDADE: Garantir que as entidades envolvidas no processo sejam realmente quem elas afirmam ser, ou seja, deve existir uma maneira de confirmação de identidade mútua;

INTEGRIDADE: Trata-se de garantir a confiabilidade dos dados, ou seja, assegurar que o dado acessado não foi alterado sem autorização. Um exemplo desta propriedade seria a utilização de *hash MD5* para garantir a integridade do arquivo;

IRRETRATABILIDADE: Esta propriedade se refere ao não repúdio de mensagens ou dados. Devido a esta propriedade, nenhuma entidade pode negar que participou de uma comunicação, sendo possível comprovar que uma mensagem veio de um remetente específico. Também é possível identificar quem acessou um determinado arquivo;

DISPONIBILIDADE: É a capacidade de conseguir acessar as informações sempre que desejado, ou seja, um sistema com alta disponibilidade oferece o serviço sempre que o usuário necessita.

Com estas informações é possível ter uma noção das propriedades e características necessárias para se construir um sistema seguro. Quando se adapta estas características e propriedades

para sistemas na nuvem é possível observar que muitas das características e propriedades básicas também são necessárias nesse ambiente (ZHOU *et al.*, 2010).

2.2.2 Armazenamento de arquivos na nuvem

O armazenamento de arquivos é um dos usos primários da computação em nuvem. Com o armazenamento na nuvem, os dados são guardados em múltiplos servidores de terceiros, ao invés dos servidores dedicados usualmente utilizados no armazenamento tradicional.

Quando está armazenando informações, o usuário acredita que está acessando um servidor único com um nome específico. Porém, na verdade este servidor não existe, sendo apenas um pseudônimo usado para referenciar um espaço virtual obtido na nuvem. Os dados do usuário podem estar guardados em vários computadores utilizados na nuvem (ITANI; KAYSSI; CHEHAB, 2009).

O armazenamento de arquivos na nuvem é uma tarefa complexa. Para se criar e especificar um modelo deste tipo devem ser ponderados diversos fatores, como o armazenamento distribuído, o desempenho do sistema, que deve ser eficiente, pois quando se fala em ambientes em nuvem, usualmente se fala em grandes escalas. Além disso, também devem ser considerados fatores como recuperação de erros, integridade dos arquivos, além das operações usuais de leitura e escrita de arquivos.

Tabela 1 – Classificação dos serviços na nuvem disponíveis. (Adaptado de Gao *et al.* (2010))

Categoria	Definição	Exemplo
Instância de Armazenamento	Armazenamento juntamente com imagens de máquinas virtuais	<i>Amazon EC2 instance storage</i>
Objeto de Armazenamento	Armazenamento de objetos binários provido na forma de Web Services. Um objeto pode ser qualquer tipo de arquivo	<i>Amazon Simple Storage Service (S3)</i>
Bloco de Armazenamento	Discos rígidos virtuais que podem ser conectados com máquinas virtuais e usados como instâncias locais	<i>Amazon Elastic Block Store</i>
Armazenamento de dados semi-estruturados	Banco de dados para guardar dados semi-estruturados com alta disponibilidade, alta escalabilidade e alto desempenho	<i>Amazon Simple DB</i>
Armazenamento de Banco de dados relacional	Servidores de banco de dados relacional em instâncias de máquinas virtuais na nuvem	<i>Amazon Relational Database Service</i>
Sistema de arquivos distribuídos	Armazenamento distribuído disponibilizado por meio de interfaces de arquivos de sistema com alta disponibilidade e alta escalabilidade	<i>Google File System</i>
Disco Online/Serviço de pasta	Armazenamento disponibilizado na forma de um disco ou pasta virtual na Internet	<i>Microsoft® OneDrive</i>

A Tabela 1 mostra uma classificação realizada por (GAO *et al.*, 2010). Nesta classificação, é possível verificar a distinção entre os tipos de serviços de armazenamento em nuvem oferecidos pelos provedores comerciais atuais. O foco deste trabalho é na categoria de “Disco Online/Serviço de pasta”.

Como pode ser observado, a classificação taxonômica dos serviços de armazenamentos na nuvem é feita levando-se em consideração diversos fatores principais, como a forma na qual o serviço será disponibilizado e as possibilidades de acesso. Também é determinante nesta taxonomia as propriedades que este serviço apresenta, como por exemplo, alta disponibilidade e a alta escalabilidade. Outro aspecto fundamental destas arquiteturas é o formato dos dados armazenados, como arquivos normais, banco de dados relacionais e semi-estruturados (ITANI; KAYSSI; CHEHAB, 2009).

Figura 2 – Arquitetura típica de um ambiente em nuvem (Adaptado de Kumar *et al.* (2012))

Uma arquitetura típica de armazenamento em nuvem pode ser vista na Figura 2. Ela é composta basicamente de um nó de controle e diversos servidores de arquivos. Todos estes dispositivos se conectam entre si, porém todo o gerenciamento dos serviços é feito pelo nó de controle, sendo que os servidores de armazenamento apenas guardam os arquivos. Normalmente, o único dispositivo com acesso à Internet é o nó de Controle. Este nó pode ser tanto um servidor dedicado para esta função ou um servidor de arquivos com uma aplicação de gerenciamento (KUMAR *et al.*, 2012).

2.2.3 Dispersão de informações

Um algoritmo de dispersão de informações é um método para dividir um arquivo F em N partes de maneira que o arquivo possa ser reconstruído utilizando um número de partes M pré definido. Estes algoritmos possuem diversas aplicações, como a transmissão eficiente de informações em sistemas distribuídos tolerantes a falhas, armazenamento de informação de maneira segura e confiável e até mesmo para comunicação entre processadores em computação paralela (SANTIS; MASUCCI, 2002).

Nos trabalhos de Rabin ([RABIN, 1989a](#); [RABIN, 1990](#)), foram propostos algoritmos para dividir um arquivo em N pedaços, de maneira que seja possível recuperar o arquivo com quaisquer M pedaços, sendo $M \leq N$.

Figura 3 – Funcionamento básico de um algoritmo de dispersão de informações (Adaptado de [Rabin \(1989a\)](#), [Rabin \(1990\)](#))

Na Figura 3 é possível observar o processo de divisão e armazenamento de arquivos de uma arquitetura típica de sistemas que utilizam a dispersão de informações. Basicamente existe a divisão do arquivo em diversas partes e posteriormente são armazenados em diversos servidores.

Figura 4 – Funcionamento básico de um algoritmo de dispersão de informações (Adaptado de [Rabin \(1989a\)](#), [Rabin \(1990\)](#)). Mesmo com as partes C e F indisponíveis, ainda é possível recuperar o arquivo.

O interessante deste algoritmo é que o arquivo consegue ser recuperado mesmo faltando algumas partes. De acordo com o tipo de algoritmo utilizado e a quantidade de partes em que o arquivo é dividido, pode-se recuperar o arquivo utilizando um número fixo de partes menor do que o número total de partes em que o arquivo foi dividido. Esta propriedade proporciona níveis elevados de disponibilidade e redundância, já que mesmo com algumas partes inacessíveis, o arquivo ainda pode ser recuperado. No exemplo, mostrado na Figura 4, foi utilizado uma

abordagem 6/4, isto é, o arquivo é dividido em 6 partes, porém com quaisquer 4 partes é possível reconstruí-lo corretamente. No caso do exemplo, as partes C e F estão indisponíveis, mas mesmo sem essas partes ainda é possível reconstruir o arquivo original. Isto é possível pois cada parte possui redundância de informações de outras partes (RABIN, 1990).

De maneira formal, um algoritmo de dispersão de informações, definido por Santis e Masucci (2002) é usado para dividir m arquivos $f_1 \dots f_m$ em um conjunto de partes $p_i \forall i \in P$, onde $P = 1, \dots, n$. Para qualquer $j = 1, \dots, m$ sendo F_j o conjunto de todos os possíveis valores do arquivo f_j e sendo $\mathbf{F}_j$ a variável aleatória associada a F_j . Para qualquer $i = 1, \dots, n$, sendo P_i o conjunto de todos os possíveis valores da parte p_i e sendo $\mathbf{P}_i$ a variável aleatória associada a P_i . Para qualquer $A = \{i_1, \dots, i_r\} \subseteq P$, sendo $P_A = P_{i_1}, \dots, P_{i_r}$. Para qualquer $j = 1, \dots, m$ é definida a estrutura de acesso A_j em P como a família de todos os subconjuntos de índices correspondentes às peças que possibilitam a reconstrução do arquivo f_j .

DEFINIÇÃO: sendo $P = 1, \dots, n$ e $(A_1, \dots, A_m)$ a estrutura de acesso para P . Um algoritmo de dispersão de informações múltiplas para $(A_1, \dots, A_m)$ é um método para dividir m arquivos em n partes de uma forma, que, para qualquer $j = 1, \dots, m$ e qualquer $A \in A_j$, é garantida a seguinte propriedade: $H(F_j|P_A) = 0$.

TEOREMA: sendo $P = 1, \dots, n$ e $(A_1, \dots, A_m)$ a estrutura de acesso para P . Sendo $A \subseteq P$ e sendo $R(A) = j : A \in A_j$. Em qualquer algoritmo de dispersão de informações múltiplas para $(A_1, \dots, A_m)$ há ao menos um índice $i \in A$, tal que: $H(\mathbf{P}_i) \geq \frac{H(\mathbf{F}_{R(A)})}{|A|}$

2.2.4 Algoritmos e códigos de dispersão de informações

Há diversos algoritmos e códigos para a dispersão de informações na literatura que utilizam abordagens distintas. Um deles é o algoritmo de Rabin (1990), que utiliza o conceito de sistemas lineares e matrizes linearmente independentes para proporcionar as propriedades de redundância desejadas no algoritmo e outro deles é o código de Reed-Solomon (WICKER; BHARGAVA, 1999). Estes algoritmos serão brevemente explicados nesta seção. Ambos utilizam o conceito de campos finitos para operarem sobre os dados. Este conceito de campos finitos será melhor explicado após esta subseção.

Código de correção de erros Reed-Solomon

O algoritmo de Reed-Solomon (WICKER; BHARGAVA, 1999) pode ser utilizado para implementar um mecanismo de dispersão de informações eficiente. Ele descreve uma maneira sistemática de criar códigos com a capacidade de detecção e correção de erros. Pela adição de t símbolos de checagem aos dados, um código RS (Reed-Solomon) consegue detectar qualquer combinação de até t símbolos errados, ou corrigir até $t/2$ símbolos.

Nó código de Reed-Solomon, os símbolos de origem são vistos como coeficientes de um polinômio $p(x)$ sobre um campo finito. A idéia então é criar n símbolos codificados a partir de k símbolos de origem pela super amostragem de $p(x)$ em $n > k$ pontos distintos, transmitir os pontos de amostragem, e utilizar técnicas de interpolação polinomial no receptor para recuperar a mensagem original.

A idéia principal deste código é conseguir representar os dados utilizando uma equação no espaço finito. Então por exemplo, caso se queira representar os inteiros 3 e 5, eles podem ser representados por meio de dois pontos (1,3) e (2,5) que se localizam em uma reta. Com isso, uma maneira de transmiti-los com redundância é enviar outros pontos desta reta, como o ponto (3,7) e (4,9). Assim, caso sejam recebidos quaisquer dois pontos, é possível construir a reta e determinar a mensagem inicial (3 e 5). Este exemplo pode ser extrapolado para mensagens de diversos tamanhos, bastando variar a equação geradora.

Algoritmo de correção de erros Rabin

O algoritmo de Rabin ([RABIN, 1989a](#)) utiliza uma abordagem distinta do código de Reed-Solomon. Ele utiliza o conceito de sistemas lineares para criar a redundância necessária. Portanto, ele consegue obter um grau de redundância maior utilizando menos espaço de armazenamento.

Analogamente aos sistemas lineares, em que dado um conjunto de equações é possível determinar as variáveis do sistema, o algoritmo de Rabin utiliza diversas equações especificadas no formato de uma matriz, onde se determina a quantidade de partes a se dividir no total (n) e a quantidade de partes necessárias (m), sendo que estes valores determinam a dimensão da matriz.

Feito isto, é necessário gerar uma matriz linearmente independente com as dimensões desejadas. Uma maneira simples de gerar este tipo de matriz é utilizar a matriz de Cauchy e de Vandermonde ([HEINIG, 1995](#)), que garante que a matriz seja linearmente independente. Em seguida, é necessário que a matriz tenha m linhas (equações) e n colunas (variáveis) garantindo que tenha apenas uma solução possível, sendo um sistema possível e determinado (SPD).

Com isso, é necessário obter os coeficientes de apenas n equações para se determinar o valor das variáveis. Portanto, para se codificar, basta multiplicar a mensagem original agrupada em blocos de n bytes pela matriz de dispersão gerando m bytes que são colocados separadamente em cada parte. Para juntar, basta multiplicar n bytes recuperados de n partes pelo inverso da matriz multiplicadora e com isso gerar novamente a mensagem original.

Como pode ser observado, este algoritmo utiliza exaustivamente multiplicação e inversão de matrizes, o que pode influenciar no seu desempenho geral, porém garante um aproveitamento de espaço eficiente.

2.2.5 Campos Finitos

Na matemática discreta, um campo finito, também conhecido como campo de *Galois* (nomeado a partir do seu criador, Évariste Galois) é um campo algébrico que contém um número finito de elementos. Os campos finitos são muito utilizados em diversas áreas, como a criptografia, correção de erros, geometria algébrica, entre outros. Neste trabalho eles são exaustivamente utilizados no processo de transformação dos dados da dispersão de informações. Eles podem ser classificados de acordo com a classificação proposta em [Jacobson \(2009\)](#):

- A *ordem*, ou número de elementos de um campo finito é da forma p^n , onde p é um número primo denominado de *característica* do campo, e n é um inteiro positivo.
- Para todo número primo p e um inteiro positivo n , existe um campo finito com p^n elementos.
- Quaisquer dois campos finitos com o mesmo número de elementos são *isomórficos*.

Neste texto, a notação a ser utilizada para os campos finitos é do formato $\mathbf{GF}(p^n)$. Em especial, serão utilizados os campos finitos $\mathbf{GF}(2^n)$, pois eles facilitam as operações utilizando elementos binários.

A aritmética dos campos finitos é diferente da aritmética de inteiros. Como existe um número limitado de elementos no campo finito, todas as operações realizadas em um campo finito resultam em um elemento dentro deste campo. Elementos de um campo finito podem ser representados como polinômios, como por exemplo, no caso de um campo finito de característica 2 ($\mathbf{GF}(2^n)$), é normal expressar elementos deste campo como números binários, com cada coeficiente do polinômio sendo representado por um bit na expressão binária correspondente. Por exemplo, as representações abaixo são equivalentes em $\mathbf{GF}(2^n)$:

- Polinomial: $x^5 + x^4 + x + 1$
- Binária: 00110011
- Hexadecimal: $H33$

A adição e a subtração são realizadas adicionando ou subtraindo os polinômios e reduzindo o resultado ao módulo da característica. Em um campo finito de característica 2, a adição módulo 2, subtração módulo 2 e a XOR são idênticas, portanto:

- Polinomial: $(x^6 + x^4 + x + 1) + (x^7 + x^6 + x^3 + x) = x^7 + x^4 + x^3 + 1$
- Binária: $01010011 + 11001010 = 10011001$
- Hexadecimal: $53h + CAh = 99h$

É importante observar que no caso de uma adição normal de polinômios, a soma iria conter um termo $2x^6$, mas este elemento se torna $0x^6$ e é descartado quando o resultado é reduzido módulo 2.

No caso da multiplicação em um campo finito, o resultado é a multiplicação módulo o polinômio irreduzível usado para definir o campo finito. O símbolo “•” é utilizado para denotar uma multiplicação em um campo finito.

Para demonstrar esta operação será utilizado o campo finito de *Rjindael* (DAEMEN; RIJMEN, 2000), que usa um campo finito de característica 2 com 256 elementos, usualmente denotado por $\mathbf{GF}(2^8)$. Ele utiliza o seguinte polinômio de redução para a multiplicação: $x^8 + x^4 + x^3 + x + 1$. Por exemplo, $53h \bullet CAh = 01h$ no campo de *Rjindael*, pois:

$$\begin{aligned} (x^6 + x^4 + x + 1)(x^7 + x^6 + x^3 + x) = \\ (x^{13} + x^{12} + x^9 + x^7) + (x^{11} + x^{10} + x^7 + x^5) + (x^8 + x^7 + x^4 + x^2) + (x^7 + x^6 + x^3 + x) = \\ x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x \end{aligned}$$

Por fim,

$$(x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x) \bmod (x^8 + x^4 + x^3 + x + 1) = 3F7Eh \bmod 11Bh = 01h.$$

Com estas operações e o campo finito de *Rjindael*, é possível obter algumas propriedades aritméticas importantes no contexto deste trabalho, como o processamento mais eficiente dos arquivos e propriedades de redundância.

2.2.6 Aplicações da dispersão de arquivos em ambientes distribuídos

Os algoritmos de dispersão de informações são utilizados em diversos contextos e em distintas aplicações, principalmente em sistemas distribuídos. Dentre estas aplicações, podem-se citar algumas na área de grades móveis, como por exemplo o trabalho de Arias e Rivera (2006), em que são utilizadas duas abordagens para a redundância de dados de radares, uma utilizando algoritmos de dispersão de informações e outra utilizando um algoritmo de replicação. A abordagem utilizando IDA (acrônimo em inglês para *Information Dispersal Algorithm*) teve um nível de confiabilidade maior e utilizava menos espaço de armazenamento. Porém, esta abordagem aumentou significativamente o tempo de resposta do sistema, o que acaba influenciando na decisão de se utilizar esta abordagem no tratamento de grades computacionais.

Ainda na área de grades, há o trabalho de Luna *et al.* (2008), em que o IDA é utilizado na proposta de uma grade composta de computadores tradicionais que possua certos níveis de segurança mais elevados, sendo adequada, por exemplo, para aplicações médicas, que naturalmente possuem requisitos de confiabilidade, integridade e confidencialidade. Por fim, outra aplicação do IDA em grades computacionais é discutida em Lu *et al.* (2010), em que é proposto um modelo de armazenamento paralelo para estes ambientes. nesse trabalho também é feita uma comparação

com técnicas de replicação tradicionais e se conclui que a abordagem utilizando IDA é mais eficiente tanto em termos de espaço de armazenamento utilizado quanto na velocidade de acesso.

Outra área de aplicação dos algoritmos de dispersão de informações é em redes *peer-to-peer* (P2P), onde é possível encontrar diversos trabalhos na literatura. Em [Augustine et al. \(2013\)](#), é proposto um algoritmo para armazenamento e recuperação de dados em um ambiente P2P altamente dinâmico com muita rotatividade entre os nós. nesse trabalho, o IDA foi utilizado para providenciar um *overhead* menor no tamanho dos arquivos em relação às técnicas de replicação tradicionais. nesse trabalho não se focou muito na segurança dos arquivos, mas sim no desempenho, espaço de armazenamento e na disponibilidade dos mesmos.

Outra aplicação do IDA em redes P2P é apresentada em [Dingledine, Freedman e Molnar \(2000\)](#), onde é proposto um sistema de armazenamento anônimo que consegue resistir às tentativas de destruir dados armazenados e garante que o dado fique disponível por um tempo pré-especificado definido pelo publicador. Ainda, em [Ashraf, Anane e Bordbar \(2012\)](#) é proposto um algoritmo de dispersão de arquivos em ambientes de redes P2P móveis que utilizam DHT e que seja eficiente, seguro e confiável para ambientes com o requisito de mobilidade. Por fim, em [Bian e Seker \(2013\)](#) é apresentado um modelo de armazenamento de arquivos em ambientes P2P que utiliza aplicações recursivas do algoritmo de dispersão, buscando altos níveis de disponibilidade, confidencialidade e segurança, de maneira que um nó nunca conhece todas as informações necessárias para conseguir obter um arquivo armazenado, sendo necessário uma série de interações entre os nós para se recuperar o arquivo. Esta é uma abordagem segura, porém esbarra no problema da infraestrutura necessária para se construir este sistema e do desempenho, que chega no máximo à taxas de acesso de 6MB/s.

2.2.7 Avaliação do armazenamento de arquivos na nuvem

A avaliação e a comparação de modelos de armazenamento de arquivos na nuvem não é uma atividade trivial, pois não há um modelo padrão para avaliar este tipo de sistema.

Um modelo é proposto por [Xiao et al. \(2010\)](#) para avaliar os níveis de segurança, disponibilidade e desempenho de acesso de sistemas de armazenamento de arquivos na nuvem. Nesse trabalho, é apresentado um modelo de avaliação de ambientes de armazenamento na nuvem, juntamente com uma análise e comparação de técnicas atuais de armazenamento distribuído em termos de segurança, disponibilidade e desempenho de acesso.

Outra abordagem, é a utilização de técnicas de avaliação de sistemas de arquivos distribuídos tradicionais, como a apresentada por [Huang e Grimshaw \(2011\)](#). nesse trabalho os autores propõem a avaliação em várias etapas:

1. Avaliar o desempenho de leitura/escrita de arquivos de tamanhos diferentes;

2. Avaliar o acesso paralelo de diversos clientes para a leitura e releitura de um mesmo arquivo;
3. Avaliar o rendimento do sistema de arquivos distribuídos em relação ao número de operações realizadas por segundo com a criação de diretórios e vários arquivos de 1 kilobyte;
4. Avaliar a transferência dos dados com e sem criptografia, e com a utilização ou não de outros algoritmos especiais utilizados no sistema de arquivos distribuído avaliado em questão, mostrando as diferenças existentes entre as situações.

Esta última abordagem avalia apenas o aspecto de desempenho desses sistemas, porém não avalia a disponibilidade e a segurança, o que necessita que se utilizem outros modelos para avaliar estes dois últimos fatores, como por exemplo o modelo proposto por [Ristov, Gusev e Kostoska \(2012\)](#), em que é apresentada uma metodologia para se avaliar a segurança de sistemas na nuvem. Essa metodologia apresentada é uma adaptação da padronização de segurança da informação proposta pela ISO 27001:2005 ([CALDER, 2009](#)) para atender os problemas e as dificuldades de um ambiente virtualizado, como é o caso da nuvem.

Por fim, uma abordagem para se avaliar a disponibilidade e confiabilidade de sistemas baseados na nuvem é discutida no livro de [Bauer e Adams \(2012\)](#). Esta é uma abordagem que utiliza diversas variáveis quantitativas para determinar o grau de disponibilidade e confiabilidade deste tipo de sistema.

2.2.8 Métricas de Desempenho e Usabilidade

Para se avaliar um sistema distribuído de dispersão de arquivos é necessário utilizar algumas métricas de desempenho e usabilidade que foram estudadas e são apresentadas nesta subseção. Um estudo de trabalhos existentes que avaliam o desempenho e a qualidade de experiência (em inglês, QoE) para soluções de armazenamento em nuvem revelou quais métricas e parâmetros são importantes para se mensurar em ambientes de teste de armazenamento em nuvem distribuído. Este estudo ajudou o desenvolvimento da plataforma FlexSky revelando as seguintes métricas mais utilizadas para avaliação do desempenho e da usabilidade:

- **Tempo de resposta:** o tempo de resposta do sistema e das operações é a métrica mais utilizada de todas ([ABU-LIBDEH; PRINCEHOUSE; WEATHERSPOON, 2010](#); [BESSANI et al., 2011](#); [MAR, 2011](#); [JAMMALAMADAKA et al., 2013](#); [ZHANG et al., 2013](#); [SCHNJAKIN et al., 2013](#)) e é utilizado em todos os trabalhos estudados. É uma métrica essencial para tais estudos. Sua unidade mais utilizada é em ms.
- **Vazão:** a vazão do sistema também é muito utilizada ([BESSANI et al., 2011](#); [SCHNJAKIN et al., 2013](#)) para avaliar o desempenho dos métodos utilizados. Esta métrica é obtida

dividindo-se a quantidade de dados transferidos pelo tempo de resposta da operação. Sua unidade mais utilizada é Mb p/ segundo (Mbps).

- **Custo de armazenamento:** esta é uma métrica econômica encontrada na literatura ([ABU-LIBDEH; PRINCEHOUSE; WEATHERSPOON, 2010](#)). Ela compreende os custos de processamento do método, juntamente com o custo para transferir e armazenar o arquivo por um período. Sua unidade mais utilizada é US\$ por Gb.

Como observado em [Casas *et al.* \(2013\)](#), tais parâmetros estão diretamente relacionados com a qualidade de experiência (QoE) do sistema. Portanto, apesar da existência de outras métricas, como a latência, segurança e disponibilidade, as três mencionadas anteriormente são normalmente suficientes para se comparar sistemas de armazenamento ([CHEN; PATTERSON, 1993](#)). Isto foi comprovado pela revisão da literatura e pelos experimentos deste projeto.

2.3 Seleção de estratégias de dispersão multinuvem de arquivos - MSSF

Esta seção aborda importantes conceitos na seleção de métodos de dispersão multinuvem, incluindo as propriedades que devem ser consideradas, parâmetros de qualidade de serviço, técnicas de seleção de serviços e as aplicações diretas destes itens no contexto de armazenamento multinuvem.

2.3.1 Métodos de seleção e avaliação de nuvens

Uma revisão de métodos de seleção e avaliação de nuvens, é apresentada em [Alabool e Mahmood \(2013\)](#). Ela mostra os métodos mais utilizadas de seleção, as métricas e os parâmetros de QoS a se avaliar e comparar para selecionar provedores de nuvem.

Tabela 2 – Técnicas de seleção de serviços em nuvem (Adaptado de [Alabool e Mahmood \(2013\)](#))

Técnica	Porcentagem de Utilização
Analytic Hierarchy Process (AHP)	20%
Fuzzy Based Approach (FBA)	15%
Feature Analysis (FA)	10%
Weight Score Method (WSM)	10%
Multi-criteria Decision Making (MCDM)	8%
Analytic Network Process (ANP)	5%
Fuzzy Analytic Hierarchy Process (FAHP)	5%
Benetic Based Approaches (GBA)	3%
Neural Network Based Approaches (NNBA)	3%

A Tabela 2 apresenta algumas destas técnicas e a utilização para seleção e classificação de serviços em nuvem. Desta forma, é possível ver que as técnicas mais utilizadas são a AHP, FBA e WSM seguida pela MCDM, que será a utilizada neste trabalho.

Uma destas técnicas denominada MCDM (sigla em inglês para tomada de decisões utilizados critérios múltiplos) é estudada por [Rehman, Hussain e Hussain \(2012\)](#) e mostra que elas são apropriadas e efetivas para selecionar serviços de nuvem.

O problema de seleção de nuvem utilizando critérios múltiplos foi posteriormente formalizado ([REHMAN; HUSSAIN; HUSSAIN, 2011](#)) em um modelo matemático rigoroso que propõe uma metodologia simples de seleção que considera apenas provedores de serviços na nuvem. Uma abordagem mais complexa de seleção MCDM é apresentada em [Le et al. \(2014\)](#). Ela utiliza lógica *fuzzy*, juntamente com técnicas de MCDM para lidar com incertezas e interdependências no processo de seleção. [Rehman, Hussain e Hussain \(2013\)](#) propôs um *framework* para selecionar automaticamente provedores de serviços na nuvem em ambientes empresariais com a necessidade de suportar políticas internas e inconsistências nos requisitos do usuário.

2.3.2 Métricas de seleção de nuvens

Quando se lida com a classificação e seleção de nuvens as métricas de avaliação são aspectos muito importantes e que devem ser considerados. Elas orientam o processo de seleção e priorizam os atributos que se deseja maximizar e minimizar.

Tabela 3 – Principais métricas encontradas na literatura de seleção de nuvens (Adaptado de [Alabool e Mahmood \(2013\)](#))

Métrica	Porcentagem de Utilização
Custo	24%
Disponibilidade e Durabilidade	19%
Desempenho	18%
Confidencialidade	15%
Reputação	10%
Confiança	7%

Uma revisão da utilização destas métricas é apresentada na Tabela 3. Nela pode-se observar que as métricas mais consideradas são: custo; disponibilidade e durabilidade; desempenho; e confidencialidade. Outras métricas, como a reputação e a confiança, que são qualitativas, também são importantes, porém devido à dificuldade de se mensurá-las de maneira efetiva elas não foram consideradas no modelo criado. Um mesmo trabalho pode apresentar mais de uma métrica, por isso a somatória das porcentagens é superior a 100%.

2.3.2.1 Custos

Os custos para se armazenar, manter e recuperar os dados é relevante, pois os custos podem influenciar e até serem os fatores decisivos para se escolher a configuração de disper-

são. Os principais custos relacionados com o armazenamento multinuvem são os custos de armazenamento e de banda de capacidade do canal de comunicação.

O custo de armazenamento é o preço que se paga para se manter os dados armazenados nos provedores por um período. O custo de banda é o preço para se transferir dados entre os provedores e o cliente. Este custo é categorizado em custo de banda de entrada e saída. O primeiro é o custo para se escrever dados no provedor e o segundo é o custo para se recuperar dados do provedor.

Os custos de banda costumam estar especificados no SLA (*Service Level Agreement*) para os maiores provedores de armazenamento na nuvem e são utilizados para precificar a oferta de serviços. Por exemplo, o Amazon S3 oferece diversas ofertas de armazenamento com preços e características distintas¹.

É importante notar que os custos de processamento para se dispersar um arquivo não são considerados neste contexto, pois todo o processo é feito localmente no cliente e se assume que o custo para utilizar recursos da máquina cliente seja irrelevante. Porém, caso seja necessário, este custo pode ser facilmente adicionado no mecanismo de seleção, basta adicionar um novo parâmetro de QoS para ser contabilizado no modelo e ele já consegue suportar.

2.3.2.2 Sobrecarga de Armazenamento

A sobrecarga de armazenamento influi diretamente no custo. Ela é uma métrica que representa a quantidade de espaço extra que o arquivo utiliza após ter sido disperso. Ela é calculada pela razão $\frac{Size_{final}}{Size_{inicial}}$ entre os tamanhos finais e iniciais. Alguns algoritmos de dispersão apresentam distintas características em relação a esta propriedade devido à natureza dos algoritmos. Ainda, o algoritmo de compressão e o nível de compressão também influenciam nesta métrica na medida em que comprime o arquivo tornando o tamanho final do arquivo menor. Apesar desta métrica ser diretamente relacionada com os custos, é importante separá-las, de forma que se torne possível analisar a influência desta propriedade separadamente em relação ao custo dos provedores.

2.3.2.3 Disponibilidade e durabilidade

Dois importantes parâmetros de qualidade de armazenamento são a disponibilidade e a durabilidade. O primeiro é definido como a razão entre (a) o tempo total que uma unidade funcional é capaz de ser usada durante um intervalo e (b) a duração total do intervalo (MILLER; VANDOME; JOHN, 2010). Este parâmetro é relacionado com a probabilidade do sistema funcionar quando requisitado. Já a durabilidade pode ser definida como “uma propriedade que garante que os dados possam ser eventualmente recuperados, porém com a possibilidade de algum atraso caso os dados estejam indisponíveis no momento (LEFEBVRE; FEELEY, 2004). Em resumo, a disponibilidade mede a probabilidade de se recuperar os dados quando necessário,

¹ <http://aws.amazon.com/pt/s3/pricing/>

enquanto a durabilidade é a propriedade que mensura a probabilidade de que nenhuma perda de dados ocorra durante um período.

Provedores de nuvem necessitam prover altos níveis de disponibilidade e durabilidade dos dados para seus usuários, mas não é raro encontrar eventos nos quais estas propriedades não são respeitadas². Uma maneira de aumentar os níveis destas propriedades é com o armazenamento multinuvem utilizando códigos de apagamento. Quando métodos deste tipo são utilizados, estas propriedades podem ser agregadas e aumentadas de acordo com as equações 2.1 e 2.2.

$$A = \sum_{i=n}^m \sum_{P \in F_i} \prod_{i \in P} A_i \prod_{j \in P^c} (1 - A_j) \quad (2.1)$$

$$D = \sum_{i=n}^m \sum_{P \in F_i} \prod_{i \in P} D_i \prod_{j \in P^c} (1 - D_j) \quad (2.2)$$

Estas equações são baseadas na função de distribuição acumulada da distribuição de Poisson Binomial (CHEN; LIU, 1997). A equação 2.1 representa a probabilidade de que, a partir de m servidores com distintos níveis de disponibilidade $A_1, A_2, \dots, A_i$, pelo menos n servidores irão funcionar quando necessário para recuperar o arquivo. A equação 2.2 representa a probabilidade de que, a partir de m servidores com distintos níveis de durabilidade $D_1, D_2, \dots, D_i$, no mínimo n servidores irão manter os dados armazenados durante um período. F_i é o conjunto de todos os subconjuntos de i provedores que podem ser selecionados a partir do conjunto total de provedores P , sendo P^c o complemento deste referido conjunto. É possível notar por meio das equações que quanto maior a relação entre m e n , mais computacionalmente complexo é computar estes valores na prática utilizando métodos tradicionais (problema combinacional). Porém, há maneiras mais eficientes para aproximar o resultado (CHEN; LIU, 1997).

2.3.2.4 Desempenho

O desempenho é uma importante métrica no contexto de métodos de dispersão multinuvem porque várias das técnicas de dispersão podem ser computacionalmente custosas. Esta métrica agrega o desempenho dos provedores e dos módulos (ZHAO *et al.*, 2013).

O desempenho do provedor é mensurado por meio da média da taxa de transferência histórica, que tem relação direta com a velocidade de armazenamento e recuperação dos provedores na nuvem (GARG; VERSTEEG; BUYYA, 2011). O desempenho dos módulos é mensurado comparando-se os módulos utilizando *benchmarks* na mesma máquina de teste, criando uma classificação linear para cada módulo considerando tamanhos de arquivo distintos. Por fim, ambas as métricas são normalizadas e inseridas no banco de dados para serem utilizadas pelo módulo de otimização.

² <http://www.infoworld.com/article/2606209/cloud-computing/162288-The-worst-cloud-outages-of-2014-so-far.html>

2.3.2.5 Confidencialidade

Esta métrica define a probabilidade de que os dados sejam acessados por entidades não autorizadas. Esta propriedade foi comentada no contexto de armazenamento em nuvem e agora será comentada no contexto de armazenamento utilizando o paradigma multinuem. Diversos fatores podem influenciar na confidencialidade dos dados, como o algoritmo de dispersão utilizado, número de partes necessárias para recuperar o arquivo, nível de criptografia e as funcionalidades de segurança dos provedores.

O algoritmo de dispersão pode oferecer níveis fracos ou fortes de confidencialidade dependendo da sua construção (LI, 2012), o que influencia no nível de segurança geral. O número de partes necessárias para recuperar o arquivo também influencia, pois quanto maior o número de partes, menos informação cada parte revela. O nível de criptografia, determinado pelo algoritmo de criptografia utilizado e o tamanho de chave também influencia no nível de confidencialidade. Do lado do provedor, a quantidade de funcionalidades de segurança, como proteção por credenciais de acesso, segurança física e segurança do canal de comunicação também influencia na confidencialidade dos provedores.

Parâmetro de QoS	Provedores	Algoritmo de dispersão	Criptografia	Compressão
Disponibilidade	X	X		
Durabilidade	X	X		
Desempenho de leitura	X	X	X	X
Desempenho de escrita	X	X	X	X
Confidencialidade	X	X	X	X
Sobrecarga de Armazenamento		X	X	X
Custo de armazenagem	X			
Custo de banda (ENTRADA)	X			
Custo de banda (SAÍDA)	X			

Tabela 4 – Influências das categorias de seleção nos parâmetros de QoS

Um resumo de todos os parâmetros de qualidade de serviço e as categorias que os influenciam são apresentados na Tabela 4.

2.3.2.6 Classificação de arquivos

Uma pesquisa preliminar para esta pesquisa revelou que os usuários esperam métodos de dispersão distintos de acordo com características de arquivo distintas. Este trabalho propõe uma

classificação baseada no padrão de acesso aos dados para mapear o armazenamento esperado pelo usuário.

Tipo de arquivo	Padrão de acesso	Parâmetros para maximizar	Parâmetros para minimizar
Em uso	muito lido e muito escrito	desempenho de leitura e escrita, disponibilidade, durabilidade	sobrecarga de armazenamento, custo de banda (entrada e saída), custo de armazenamento
Ativo	muito lido e pouco escrito	Desempenho de escrita, disponibilidade, durabilidade, desempenho de escrita	sobrecarga de armazenamento, custo de banda (entrada e saída), custo de armazenamento
Backup-Ativo	pouco lido e muito escrito	disponibilidade, durabilidade, desempenho de leitura	sobrecarga de armazenamento, custo de banda (entrada), custo de armazenamento, custo de banda (saída)
Backup-Inativo	pouco lido e pouco escrito	durabilidade, disponibilidade, desempenho de escrita	sobrecarga de armazenamento, custo de armazenamento, custo de banda (entrada e saída)

Tabela 5 – Resumo dos tipos de arquivos e suas características

A classificação utilizada neste trabalho é apresentada na Tabela 5 e mostra quatro tipos de arquivos classificados de acordo com seus padrões de acesso e parâmetros de otimização para maximizar e minimizar ordenados pela importância (do mais importante para o menos importante) no processo de dispersão.

Para determinar a classificação dos arquivos, os parâmetros de entrada podem ser fornecidos pelo usuário ou inferidos de acordo com as propriedades e atributos do arquivo (MESNIER *et al.*, 2004).

2.4 Considerações finais

Este capítulo apresentou os conceitos necessários para o melhor entendimento deste trabalho de pesquisa. Ele incluiu conceitos básicos para se entender armazenamento de arquivos e seleção de serviços em ambientes multinuvem. Com este arcabouço fica mais prático entender os trabalhos relacionados e a contribuição deste projeto de pesquisa.

TRABALHOS RELACIONADOS

3.1 Considerações Iniciais

A revisão sistemática realizada na literatura de sistemas de arquivos distribuídos revelou algumas abordagens anteriores utilizadas por pesquisadores da área visando sistemas de armazenamento de arquivos mais eficientes e que contenham certas propriedades essenciais. Este capítulo apresenta esta revisão e no final uma análise comparativa entre todos os trabalhos desenvolvidos e o projeto de pesquisa em questão.

3.2 Trabalhos relacionados com a FlexSky

Esta seção apresenta e sumariza os trabalhos existentes na literatura que propõem alguma arquitetura para dispersão de arquivos em ambientes multinuvem.

3.2.1 *High-availability Integrity Layer (HAIL)*

Em HAIL (*High-availability Integrity Layer*) (BOWERS; JUELS; OPREA, 2009) é proposto um sistema que pretende garantir a integridade e disponibilidade dos dados utilizando uma abordagem semelhante aos sistemas RAID (CHEN *et al.*, 1994), onde ao invés dos discos são utilizados os serviços de armazenamento na nuvem. O HAIL utiliza um conjunto de protocolos criptográficos juntamente com códigos de apagamento (em inglês, *erasure-codes*) e provas de recuperação. Com isso, é possível criar uma camada de abstração para proteger a integridade dos dados, mesmo que estas sejam invadidas e corrompidas por um possível atacante. Porém, o HAIL apresenta algumas limitações importantes: somente trabalha com dados estáticos, requer que os servidores executem código e não se preocupa com a confidencialidade e a privacidade dos dados armazenados.

3.2.2 *Redundant Array of Cloud Storage - RACS*

O sistema RACS (*Redundant Array of Cloud Storage*) ([ABU-LIBDEH; PRINCEHOUSE; WEATHERSPOON, 2010](#)) utiliza uma abordagem similar às encontradas nos sistemas RAID nível 5 ([CHEN *et al.*, 1994](#)) para realizar a replicação dos dados. Porém, este sistema se preocupa muito em minimizar os custos das falhas “econômicas”, de acordo com a definição do autor do trabalho, sem muita atenção com os aspectos de confidencialidade, segurança e privacidade dos dados.

3.2.3 *iDataGuard*

Os autores do *iDataGuard* ([JAMMALAMADAKA *et al.*, 2013](#)) descrevem a aplicação como um sistema que combate a heterogeneidade dos provedores de armazenamento de dados, que provê mecanismos de segurança para os dados e ainda possibilite a pesquisa sobre os dados cifrados. Ele propõe um *middleware* que pode ser executado na máquina cliente ou alternativamente em um servidor *proxy* considerado seguro que consiga abstrair os detalhes de cada provedor de serviço de armazenamento criando uma interface abstrata que comunique com os provedores de maneira transparente para o usuário. Apesar de possibilitar a confidencialidade e a busca nos arquivos criptografados, o que é uma funcionalidade muito interessante, este sistema não provê nenhum meio de redundância e de disponibilidade, ele apenas armazena os dados de maneira segura em ambientes considerados inseguros.

3.2.4 *DepSky*

Outro sistema investigado é o *DepSky* ([BESSANI *et al.*, 2011](#)), que é descrito pelos autores como uma nuvem virtual que engloba outras nuvens buscando ampliar o nível de disponibilidade, integridade e confidencialidade de dados. O *DepSky* utiliza o conceito de partilha de segredos para garantir a confidencialidade dos dados. Este conceito é similar ao que será utilizado neste trabalho de mestrado. A integridade dos dados é feita utilizando checagens de integridade e a utilização de *locks* durante as operações de escrita e leitura. Para garantir a disponibilidade, são utilizados sistemas de quóruns bizantinos de disseminação ([MALKHI; REITER, 1998](#)), o que requer um número significativo de servidores para tolerar as falhas e custa um grande *overhead* no armazenamento.

3.2.5 *Secured Virtual Diffused File System - SVDFS*

O autor do SVDFS (*Secured Virtual Diffused File System*) ([MAR, 2011](#)) propõe uma abordagem muito semelhante à que será utilizada neste presente trabalho, utilizando o IDA (algoritmo de dispersão de dados) para prover integridade, confiabilidade e confidencialidade dos dados de maneira eficiente, nas operações de escrita e leitura sequenciais e aleatórias. Porém, este sistema necessita obrigatoriamente um servidor específico para gerenciar os metadados, o

que proporciona um ponto de falha caso este servidor seja comprometido e amplia os custos de se implementar esta solução. Ainda, esta arquitetura não proporciona uma interface uniforme e transparente para interagir com os provedores de armazenamento.

3.2.6 CloudSafe

Outro trabalho encontrado na literatura que se aproxima desta proposta é o *CloudSafe* (ZHANG *et al.*, 2013), que utiliza o código de *Reed-Solomon* juntamente com a criptografia para prover a disponibilidade e a segurança dos dados. Esta abordagem é interessante pois não necessita de uma infraestrutura centralizada (servidor de registro) e integra diversos provedores de serviços de armazenamento na nuvem. Com quatro servidores é possível conseguir uma eficiência de armazenamento de 1,33x. Porém, esta abordagem possibilita apenas a falha de um servidor, sem possibilidade de personalizar a redundância desejada. Os autores também mostram que para arquivos superiores a 10Mb, o tempo para se criptografar e descriptografar os arquivos é um fator limitante importante.

3.2.7 SCFS

SCFS (BESSANI *et al.*, 2014) é outra implementação de armazenamento multinuvel. Ele foca em propriedades e estratégias de compartilhamento de arquivos. Apesar de considerar o compartilhamento de arquivos, ele utiliza uma arquitetura centralizada e não oferece nenhuma flexibilidade para se testar estratégias de dispersão. O usuário fica preso aos mecanismos de dispersão já implementados.

3.2.8 NubiSave

NubiSave (SPILLNER; MULLER; SCHILL, 2013) possibilita estender parte de seu fluxo de trabalho interno. Porém, ele não suporta a execução de experimentos em lote e a variação de parâmetros para os módulos de dispersão. Isto limita as possibilidades dos módulos adicionais para a ferramenta, pois ele só consegue ler e escrever diretamente nos fluxos de escrita e leitura, sem a possibilidade de suportar parâmetros nos módulos.

3.2.9 Cloud-RAID

Cloud-RAID (SCHNJAKIN *et al.*, 2013) também utiliza a abordagem RAID para superar as limitações do armazenamento na nuvem com um único provedor da mesma maneira que o RACS. Porém, o Cloud-RAID oferece suporte para provedores de armazenamento existentes e possui propriedades de segurança.

3.2.10 Bonfire

Bonfire (HUME *et al.*, 2012) é uma plataforma completa de testes que permite o estudo de propriedades de armazenamento multinuvel. Ele utiliza um ambiente completo composto de sete centros de dados distribuídos geograficamente e provê mecanismos de teste controlados que possibilitam testar diversas estratégias de dispersão e de seleção. Apesar de ser altamente flexível e fácil de usar, ele está apenas disponível para grupos de pesquisas restritos e é fechado ao público.

3.2.11 Diferenciais deste trabalho

Por fim, diante desta revisão dos trabalhos com propostas semelhantes a este projeto de mestrado é possível identificar algumas funcionalidades desejadas neste tipo de sistema. Muitos trabalhos utilizam criptografia para prover a confidencialidade dos dados, porém esta abordagem não é muito eficiente, visto que acaba exigindo uma grande quantidade de recursos computacionais, o que pode inviabilizar o uso destas ferramentas ao usuário tradicional.

A utilização de criptografia não se mostra tão necessária quando se trabalha com algoritmos de dispersão que consigam garantir a confidencialidade do arquivo (caso não se tenha o mínimo de partes e a matriz de decodificação, o que, pode ser explorado neste tipo de sistema). Ainda, nenhum dos trabalhos possibilitaram flexibilizar o local de armazenamento dos metadados, sejam eles chaves ou matrizes de codificação. Todos os trabalhos ou necessitavam de uma infraestrutura (como um servidor de registro) para gerenciar os metadados, ou guardavam estas informações apenas na máquina do cliente, sem a possibilidade de manter estas informações em mídias de armazenamento removíveis ou em outros locais escolhidos pelo usuário.

Outra característica interessante neste tipo de sistema é a possibilidade de acesso aleatório ao arquivo, uma característica que está presente em apenas alguns sistemas propostos e que é permitida com a utilização da dispersão de informações (IDA). Por fim, a possibilidade de indexar os dados armazenados possibilitando a busca destes com sua confidencialidade preservada também é uma importante funcionalidade nestes sistemas.

Ainda, é importante avaliar nestes sistemas a necessidade de executar códigos ou aplicações no lado do servidor de armazenamento, o que inviabiliza integrar as soluções desenvolvidas aos ambientes de armazenamento comerciais existentes, já que não se tem a disponibilidade de executar aplicações ou códigos nestes servidores. É desejável então, que estes sistemas de armazenamento possibilitem criar um ambiente virtual transparente para o usuário, objetivando abstrair e integrar os serviços de armazenamento utilizando apenas as funcionalidades nativas disponibilizadas, como por exemplo o envio e a recuperação de arquivos.

Na Tabela 6 é possível visualizar as principais características dos trabalhos relacionados estudados. As características avaliadas estão especificadas na lista abaixo:

- (A) Integridade;
- (B) Confidencialidade;
- (C) Nível de confidencialidade flexível (usuário pode escolher diversos níveis de segurança para cada arquivo);
- (D) Disponibilidade;
- (E) Nível de disponibilidade flexível (usuário pode escolher a disponibilidade desejada para cada arquivo, inclusive a ausência desta propriedade);
- (F) Flexibilidade na escolha do local de armazenamento dos meta-dados;
- (G) Suporte a heterogeneidade (suporte fácil às interfaces de várias nuvens de armazenamento);
- (H) Flexibilidade de alterar o algoritmo de dispersão utilizado;
- (I) Flexibilidade de se alterar/criar os módulos de pré/pós processamento (compressão, criptografia, etc);
- (J) Suporte a seleção da estratégia de dispersão (criar e variar seletores);
- (K) Suporte a execução de experimentos em lote automatizado;
- (L) Não necessita de infraestrutura adicional (utiliza apenas os provedores e o cliente);
- (M) Código aberto.

Tabela 6 – Características dos trabalhos semelhantes

Sistema	A	B	C	D	E	F	G	H	I	J	K	L	M
(BOWERS; JUELS; OPREA, 2009)	✓	-	N	✓	-	-	✓	-	-	-	-	-	✓
(ABU-LIBDEH; PRINCEHOUSE; WEATHERSPOON, 2010)	✓	-	N	✓	✓	-	✓	-	-	-	-	-	✓
(JAMMALAMADAKA <i>et al.</i> , 2013)	✓	✓	-	-	N	-	✓	-	-	-	-	✓	✓
(BESSANI <i>et al.</i> , 2011)	✓	✓	✓	✓	✓	-	✓	-	-	-	-	-	✓
(MAR, 2011)	✓	✓	-	✓	✓	-	-	-	-	-	-	-	✓
(ZHANG <i>et al.</i> , 2013)	✓	✓	-	✓	-	-	✓	-	-	-	-	✓	✓
(SCHNJAKIN <i>et al.</i> , 2013)	✓	✓	-	✓	-	-	✓	-	-	-	-	✓	✓
(SPILLNER; MULLER; SCHILL, 2013)	✓	✓	✓	✓	✓	-	✓	✓	✓	-	✓	✓	✓
(BESSANI <i>et al.</i> , 2014)	✓	✓	-	✓	-	-	✓	-	-	-	-	-	✓
(HUME <i>et al.</i> , 2012)	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	-	-
FlexSky	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

N - Não se Aplica

3.3 Trabalhos relacionados com o MSSF

Esta seção apresenta os trabalhos relacionados ao problema de seleção de estratégias de dispersão multinuvem de arquivos.

Há diversas abordagens para se agregar provedores de nuvem. Geralmente, elas permitem apenas um conjunto restrito de opções, forçando o usuário a tomar decisões técnicas manualmente ou restringindo as propriedades disponíveis aos mesmos.

Uma variante mais complexa é o problema de seleção de armazenamento multinuvem, que pode ser formalizado como um problema de seleção de composição de *web services*. [Zeng et al. \(2004\)](#) utiliza o problema clássico de otimização da mochila para formalizar este tipo de problema de seleção de composição de serviços levando em conta atributos de qualidade de serviço (QoS). Neste trabalho também são apresentadas algumas técnicas que podem ser utilizadas para resolver o problema de otimização. Uma destas técnicas para solucionar problemas de otimização é baseada no conceito de programação linear inteira. [Ruiz-Alvarez e Humphrey \(2012\)](#) também propõe utilizar um resolvidor linear inteiro para descobrir a melhor maneira de selecionar provedores de armazenamento na nuvem. Este trabalho mostra, por meio de experimentos que esta técnica é confiável e escalável neste contexto. [Ruiz-Alvarez e Humphrey \(2011\)](#) também propõe um esquema XML para descrever as entidades de armazenamento e um sistema de correspondência que utiliza requisitos do usuário para selecionar a melhor estratégia de armazenamento levando em conta apenas os atributos de custo e de desempenho. Este trabalho se limita na seleção dos provedores que serão utilizados e não comenta sobre os outros aspectos, como o nível de redundância, segurança, algoritmo de dispersão, entre outros.

Apesar de todos estes trabalhos oferecerem interessantes soluções para o problema de seleção de estratégia de armazenamento na nuvem, eles não são adequados para ambientes multinuvem, nos quais vários provedores devem ser selecionados juntamente com uma estratégia de dispersão que satisfaça os requisitos do usuário. Uma interessante revisão considerando ambientes multinuvem é apresentada em [Petcu \(2014\)](#). Este trabalho também propõe uma taxonomia multinuvem juntamente com problemas e requisitos a se superar quando se cria um *middleware* multinuvem. Ele oferece uma bom arcabouço de conceitos e técnicas para a seleção de serviços gerais multinuvem, porém não estuda o armazenamento multinuvem.

Feita esta introdução, o único trabalho encontrado que deve ser ressaltado nesta monografia é o *Scalia*, que atua em ambientes de armazenamento multinuvem.

3.3.1 *Scalia*

Scalia ([PAPAIOANNOU; BONVIN; ABERER, 2012](#)) propõe estudar técnicas de seleção multinuvem que consigam selecionar automaticamente uma estratégia de dispersão sem a intervenção do usuário. Ele utiliza dados históricos, padrões de acesso dos arquivos e o tempo de vida dos arquivos para otimizar continuamente a disposição dos dados utilizando múltiplos

provedores de armazenamento na nuvem. Porém, apesar de ser possível estudar a seleção multinuvem, ele não oferece suporte à seleção de estratégias de dispersão e seleção, apenas dos provedores que serão utilizados. Ainda, ele considera apenas alguns parâmetros de QoS dos provedores, como o custo, disponibilidade e durabilidade, desconsiderando outros atributos importantes, como a segurança por exemplo. Apesar de ser complexo, é atualmente a única ferramenta que temos conhecimento que permite testar outras estratégias de seleção que esteja disponível ao público. Este trabalho também não apresenta experimentos que mostrem a robustez desta abordagem de seleção para verificar sua adequação em ambientes reais.

3.3.2 Diferenciais deste trabalho

A contribuição deste trabalho neste problema de seleção se difere dos trabalhos relacionados ao prover um mecanismo de seleção que considera diversas estratégias de dispersão, provedores e módulos de transformação (criptografia, compressão). Ele ainda combina técnicas de programação linear inteira com um sistema especialista baseado em regras para possibilitar uma maneira simples e fácil de interação com o usuário.

3.4 Considerações Finais

Neste capítulo foram discutidos os trabalhos relacionados e abordagens anteriores buscando resolver os problemas alvos deste projeto de pesquisa. Também foi apresentada uma análise comparativa entre todos buscando mostrar os diferenciais deste trabalho em relação aos demais encontrados na literatura.

DESENVOLVIMENTO DO TRABALHO

4.1 Considerações Iniciais

Este capítulo apresenta o desenvolvimento do projeto de mestrado, as contribuições, a metodologia de pesquisa que foi seguida, a modelagem do sistema e resultados obtidos com o projeto.

4.2 Contribuições do projeto de pesquisa

O projeto de pesquisa apresenta duas contribuições principais. A primeira é uma ferramenta para testes automatizados de técnicas de dispersão de arquivos em ambientes heterogêneos multinuvem e a segunda é uma técnica de seleção de estratégias de dispersão de arquivos utilizando vários provedores de nuvem ao mesmo tempo.

A FlexSky contribui para facilitar o estudo de novas técnicas de dispersão no contexto multinuvem. Toda sua arquitetura foi pensada buscando a facilidade de integração e adição de funcionalidades. Devido a isto, ela foi pensada de maneira modular e baseada em interfaces de fácil entendimento. Ainda, ela possibilita o fácil planejamento e execução de experimentos reprodutíveis.

Já o MSSF é uma das primeiras abordagens que atua diretamente na seleção de estratégias para dispersão de arquivos em ambientes multinuvem levando em consideração atributos importantes de QoS. Ele utiliza uma abordagem baseada em tomada de decisões multi-critério e programação linear inteira criando um modelo de otimização que resolve o problema de seleção de estratégias de seleção.

A FlexSky foi pensada para facilitar a utilização por pesquisadores, mas também pode ser utilizada por usuários leigos. Já que com ela não será necessária nenhuma infraestrutura adicional, sendo que qualquer usuário poderá apenas com uma aplicação montar sua própria

nuvem virtual privada e testar novas funcionalidades. Favorecer e incentivar o armazenamento de dados com segurança e redundância também é uma proposta deste modelo, em que se deseja popularizar este tipo de atividade entre usuários que não possuam muita familiaridade com o armazenamento de dados na nuvem. Com isto, os usuários finais podem agregar diversos serviços de armazenamento gratuito de arquivos na nuvem de maneira a somar os espaços e as taxas de transferências disponibilizadas por estes em uma grande nuvem virtual, mais segura e mais confiável do que os serviços existentes.

É possível sumarizar o presente projeto de pesquisa nas seguintes contribuições:

- Flexibilizar a adição de novas funcionalidades

O sistema possibilita a fácil implantação de novos módulos para estudar o armazenamento de arquivos em ambientes multinuvm. Isto facilita o processo, que previamente era muito complexo e pouco intuitivo, o que levava pesquisadores a perderem um tempo grande nesta tarefa ao invés de focar no principal, que é a nova funcionalidade a ser estudada.

- Seleção de estratégias de dispersão em ambientes multinuvm

Uma contribuição para o estado da arte está na seleção de estratégias multinuvm para dispersão de arquivos. Neste projeto é apresentado um método eficaz e rápido para a seleção destas estratégias baseado em programação linear inteira.

- Utilizar provedores de armazenamento existentes

O sistema possibilita abstrair os diversos provedores de armazenamento existentes em uma interface uniforme, de maneira que para o usuário vai aparentar que ele está utilizando apenas um provedor, quando na realidade ele pode estar utilizando vários deles. Também foi desenvolvida uma classe abstrata em que se pode facilmente implementar interfaces com novos provedores utilizando as funções disponibilizadas por estes.

- Flexibilizar o nível de disponibilidade e espaço de armazenamento desejado pelo usuário

O sistema possibilita a integração com os provedores de armazenamento utilizando a dispersão de arquivos. Por isso, o usuário poderá escolher a quantidade de partes que o arquivo será dividido e a quantidade de partes necessárias para recuperá-lo, possibilitando diversos níveis de disponibilidade e de aproveitamento do espaço de armazenamento.

- Flexibilizar o nível de segurança desejado pelo usuário

O usuário tem a opção de escolher o nível de segurança desejado para cada arquivo armazenado. Caso ele queira garantir que nenhuma entidade não autorizada tenha acesso ao seu arquivo, ele pode optar pela dispersão e armazenamento dos metadados necessários para acessar o arquivo em algum local seguro e controlado, como uma mídia removível que apenas ele próprio tenha acesso físico. Os metadados são armazenados em um arquivo, sendo possível o armazenamento destes em diversos locais e inclusive sua dispersão.

- Automatizar a execução de experimentos reproduzíveis

O sistema tem suporte para realização de testes automatizados e reproduzíveis, por meio da configuração de cenários com replicações para aumentar a acurácia dos resultados do experimento.

4.3 Modelo de armazenamento de arquivos multinuvem - FlexSky

Uma das contribuições do presente projeto de mestrado é um modelo flexível para realização de experimentos no armazenamento de arquivos em ambientes multinuvem. O arquivo poderá ser enviado para diversas nuvens de maneira redundante possibilitando atingir níveis adequados de disponibilidade, segurança, desempenho e privacidade.

4.3.1 Arquitetura

Os objetivo da arquitetura FlexSky é prover um ambiente de desenvolvimento fácil para armazenamento multinuvem baseado em módulos que podem ser combinados e comparados entre si. A Figura 6 apresenta o fluxo de trabalho principal e a interação entre os módulos.

Primeiro, o usuário pode executar a aplicação por meio de uma interface gráfica (Figura 5) ou configurar um experimento em lotes utilizando o módulo “Experiment Planner” (Figura 6-passo 1b). Após, ambos módulos se comunicam com o módulo gerenciador “Manager”, o qual decide baseado nas configurações de dispersão definidas se será utilizado o seletor de dispersão (Figura 6-passo 3) ou irá utilizar diretamente os parâmetros obtidos. Assim, será executado o módulo “Blob Storage” para realizar a dispersão desejada (Figura 6-passo 4). Após este passo, será executado o módulo “Metadata Storage” para atualizar a base de metadados (Figura 6-passo 5). O módulo “Logger” monitora e grava todas as operações de todos os módulos, o que torna possível mensurar o desempenho individualmente para cada módulo.

Uma vez apresentado o funcionamento geral e a interação entre os módulos, a seguir são discutidos os propósitos de cada módulo:

- **Módulo *Experiment Planner***

Este módulo auxilia pesquisadores na criação e execução de experimentos. Ele se baseia em um conjunto de cenários (padrões de acesso, carga de trabalho, tipo de arquivos, replicação, entre outros) com as opções de armazenamento. Este módulo irá executar automaticamente os cenários desejados, para que o usuário possa executar experimentos em lote sem supervisão adicional;

- **Módulo *User Interface***

Esta é a interface gráfica para o usuário interagir com o sistema. Este módulo armazena

Figura 5 – FlexSky - Interface gráfica

Figura 6 – FlexSky - Arquitetura geral

diversos parâmetros de usabilidade além de diversos tipos de interfaces gráficas para lidar com armazenamento multinuvem;

- **Módulo *Manager***

Este é o módulo orquestrador do sistema. É o responsável por gerenciar e sincronizar todos os outros módulos e as operações de armazenamento garantindo consistência ao longo das etapas;

- **Módulo *Metadata Storage***

Este módulo é responsável por gerenciar os metadados dos arquivos para que o sistema

consiga recuperá-los sem erros. Este é um módulo flexível construído com base em requisições ao banco de dados que pode ser flexibilizado para suportar diversos SGBDs (Sistema gerenciador de banco de dados) e arquiteturas que armazenam metadados;

- **Módulo *Blob Storage***

Este módulo é responsável por armazenar os *blobs* (*inglês, binary large objects*) utilizando a configuração de dispersão desejada. Neste módulo são apresentados os conceitos de *drivers, stream pipelines e storage options*;

- **Módulo *Dispersal Selector - MSSF***

Este módulo é responsável por selecionar a configuração de armazenamento mais apropriada de acordo com as características do arquivo e comportamento do usuário. Este módulo, por ter sido base para uma das contribuições do projeto de mestrado, possui uma seção dedicada em cada capítulo da dissertação;

- **Módulo *Logger***

Este módulo é o responsável por rastrear e gravar todas as atividades do sistema, incluindo tempos de requisição (armazenamento, recuperação, entre outros). Ele permite uma avaliação mais profunda dos experimentos executados.

4.3.2 Módulo *Experiment Planner*

O planejador de experimentos foi desenvolvido para permitir a execução em lotes dos experimentos sem a supervisão do usuário. Ele também possibilita realizar experimentos reproduzíveis.

Figura 7 – FlexSky - Arquitetura do *Experiment Planner*

A Figura 7 mostra a arquitetura deste componente. O usuário submete um ou vários cenários para serem executados durante o experimento (passo 1). então, o módulo *scenario parser* obtém todos os experimentos a partir dos arquivos de cenários e os enviam ao módulo *experiment executor*. Posteriormente, o módulo de execução de experimentos cria as tarefas e

as envia para o módulo de gerenciamento (passo 2) de acordo com a quantidade de repetições configuradas. Assim que todos os experimentos são finalizados (passo 3), o executor encerra a aplicação.

Código-fonte 1: Exemplo de um cenário XML

```

1 <scenario name="backupOnly">
2   <repeat>10</repeat>
3   <db resetOnStart="true">pathToDB</db>
4   <log_db>pathToLogDB</log_db>
5   <providerList>
6     <account name="box">
7       <type>driver.WebDavDriver</type>
8       <path>https://dav.box.com</path>
9       <parameters>
10        <param key="user" value="ra"></param>
11        <param key="pass" value="un"></param>
12      </parameters>
13    </account>
14    <account name="4shared">
15      <type>driver.WebDavDriver</type>
16      <path>https://webdav.4shared.com</path>
17      <parameters>
18        <param key="user" value="ra"></param>
19        <param key="pass" value="un"></param>
20      </parameters>
21    </account>
22  </providerList>
23  <operationList>
24    <operation action="upload">
25      <file>
26        <path>/home/flexsy/document.pdf</path>
27        <revision>1</revision>
28        <type>text</type>
29        <options>
30      <filePipeline>
31        <pipe name="pipe.PipeTest"></pipe>
32      </filePipeline>
33      <slicePipeline>
34        <pipe name="pipe.PipeTest"></pipe>
35      </slicePipeline>

```

```

36 <parameters>
37   <param key="p1" value="v1"></param>
38   <param key="p" value="v2"></param>
39 </parameters>
40 </options>
41 </file>
42 <totalParts>2</totalParts>
43 <reqParts>1</reqParts>
44 <idaMethod>RabinIDA</idaMethod>
45 </operation>
46 </operationsList>
47 </scenario>

```

Um exemplo de cenário é apresentado no Código-fonte 1. Ele mostra um cenário básico com dois provedores de armazenamento (Box¹ e 4Shared²) e uma operação de envio que utiliza os dois pipelines (discutido em detalhes na Seção 4.3.3. É possível notar que o número total de partes dispersas é dois, a quantidade de partes necessárias é um e o método de dispersão escolhido é o Rabin.

4.3.3 Módulo Flexible Blob Storage

Para ser um módulo de armazenamento flexível é preciso que o módulo satisfaça três características principais: (1) facilidade de desenvolvimento, (2) robustez e (3) intercomunicação. As Figuras 8 e 9 mostram o fluxo deste módulo.

Figura 8 – FlexSky - Arquitetura interna do Flexible Blob Storage (Parte 1)

¹ www.box.com

² www.4shared.com

Este módulo é o principal de toda a arquitetura e é responsável pela maior parte da operação de dispersão. Primeiro, o módulo necessita ser configurável e ajustável para suportar diversas tecnologias de armazenamento, como armazenamento local, *WebDav*, *FTP*, *SCP*, *Web Services*, APIs proprietárias e outras tecnologias. Isto foi obtido com o conceito de *driver* de armazenamento, que é uma interface comum que oferece funções básicas de armazenamento, como as operações “put”, “get”, “delete” e “list”. Com isso, para suportar uma nova tecnologia de armazenamento, um usuário precisa apenas implementar estes métodos e então, o sistema já suportará a nova tecnologia (Figura 9-6)

Código-fonte 2: Java Pipe Process Interface

```
1 public interface IPipeProcess {  
2     public abstract void process(InputStream io, OutputStream os, HashMap<String  
    ,String> parameters);  
3     public abstract void reverseProcess(InputStream io, OutputStream os, HashMap  
    <String,String> parameters);  
4 }
```

O módulo requer uma estrutura que possa suportar facilmente o processamento de arquivos pré-dispersão e pós-dispersão, como criptografia, compressão e verificações de consistência. Antes da FlexSky, se o usuário desejasse investigar quaisquer destas técnicas em um ambiente real, ele precisaria escolher uma aplicação de dispersão multinuvem, estudá-la profundamente para entender sua estrutura e então implementar uma funcionalidade desejada entre os fluxos da aplicação. Isto normalmente aumenta a complexidade para criar tais protótipos. Com o novo conceito de “Stream Pipeline”, o usuário precisa apenas criar uma estrutura simples que implemente duas funções básicas (“process” and “reverse process”) que irão ler e escrever a partir de uma estrutura padronizada de parâmetros (como pode ser visto no Código-fonte 2). Então, a plataforma irá adicionar os módulos ao “file pipeline” (Figura 8-passo 1) e ao “slice pipeline” (Figura 8-passo 3) conforme necessário.

O módulo “file pipeline” contém as funcionalidades que utilizam ou modificam o arquivo original antes do processo de dispersão. Este módulo consegue fazer qualquer operação, como por exemplo a criptografia ou compactação dos arquivos. Após o processo de dispersão (Figura 8-passo 2), as partes são enviadas para o módulo “slice pipeline”, que tem comportamento similar ao módulo “file pipeline” porém realiza as operações nas partes do arquivo. Por fim, após passar por este módulo, estas partes são dispersas e enviadas aos provedores de armazenamento (Figura 9-4,5,6,7). Ao utilizar estas estruturas, o usuário pode facilmente criar e testar novas funcionalidades em ambientes multinuvem distribuídos.

O fluxo de trabalho deste módulo pode ser sumarizado nas seguintes etapas:

1. O arquivo inicial passa pela lista de módulos “file pipeline”, em que será utilizado/transformado pelos módulos; (Figura 8 - Passo 1)

Figura 9 – Flexible Blob Storage Internal Architecture (Part 2)

2. Após as transformações ele é enviado ao módulo dispersor que irá dividi-lo em diversas partes; (Figura 8 - Passo 2)
3. Cada uma das partes irá passar pelos módulos “slice pipeline”, em que serão utilizadas/-transformadas pelos módulos desta etapa; (Figura 8 - Passo 3)
4. Depois de transformadas, as partes serão enviadas ao módulo *Uploader* para serem enviadas aos servidores de nuvem; (Figura 9 - Passo 4)
5. O módulo *Uploader* irá recuperar os dados dos provedores de armazenamento que irão receber as partes; (Figura 9 - Passo 5)
6. Após isso, serão instanciados os *Drivers* para cada provedor de nuvem e serão enviadas as partes; (Figura 9 - Passo 6)
7. Por fim, os metadados do arquivo e da localização das partes serão armazenados em um banco de dados de metadados para serem recuperados no futuro. (Figura 9 - Passo 7)

Por brevidade, nesta monografia é mostrado apenas o fluxo de dispersão. Porém as operações dos módulos “pipeline” são aplicadas nas operações de dispersão e recuperação.

4.3.4 Módulo Logger

Para permitir a comparação entre diversas estratégias de armazenamento disperso, foi criado um módulo que grava as operações realizadas no sistema junto com tempos de respostas e outras variáveis interessantes. Os resultados de um experimento podem ser recuperados ao

acessar a base de *logs* utilizando simples comandos SQL, possibilitando a comparação dos resultados.

Este módulo também é customizável, o que permite um pesquisador criar novas métricas para serem avaliadas. Atualmente o tempo de resposta de cada módulo, a vazão e a sobrecarga no tamanho de armazenamento são gravados (c.f. seção 2.2.8).

4.3.5 Modelagem

Toda a modelagem do sistema, ressaltando às ações da plataforma, as classes e pacotes necessários para o funcionamento são apresentadas nesta subseção.

As figuras seguintes apresentam diagramas de Classe UML simplificado mostrando os principais módulos e classes do sistema. Em especial, pode ser visto a clara divisão entre as funcionalidades de cada módulo. Há um módulo denominado “driver” que é responsável por criar uma camada de abstração para os locais de armazenamento. Também existe o módulo “dispersal”, responsável pela implementação dos algoritmos de dispersão do sistema. Outro módulo importante é o “management”, responsável pela gestão e orquestração dos outros módulos do sistema e do sistema integralmente. Outro módulo importante é o “database”, responsável por toda a gestão dos metadados, espaços de armazenamentos, credenciais e criptografia dos metadados. Por fim, é apresentado o módulo “data”, responsável por caracterizar os objetos e os dados que irão operar no sistema e o pacote “utils”, que contém classes utilitárias, como geradores de números, algoritmos de criptografia, campos finitos e utilidades em geral.

A Figura 11 mostra um diagrama de classes UML simplificado do módulo “Blob Storage”. Nele é possível visualizar a interação entre as interfaces *IPipeProcess*, *IDriver*, *IDecoderIDA* e *IEncoderIDA*. Elas são os principais elementos deste módulo. Caso se queira adicionar suporte a novas tecnologias basta implementar uma destas interfaces e o sistema já irá automaticamente suportar a nova funcionalidade. Isto torna a arquitetura modular e fácil de ser personalizada.

Os tipos de dados do sistema são apresentados no diagrama da Figura 11. Nele, é possível visualizar a estrutura de dados para contas em provedores (“DataAccount”), para arquivos inteiros (“DataFile”), para partes de arquivos (“DataSlice”) e a estrutura comum “StorageOptions”, que armazena os pipelines e parâmetros adicionais. Esta estrutura é compartilhada entre todos os módulos, conforme destacado na Figura 8.

A gestão dos metadados, que inclui o gerenciamento das partes dos arquivos e arquivos armazenados é feita pelas classes que são apresentadas na Figura 12. Cada uma destas classes é responsáveis pelas operações no banco de dados relacionadas ao seu tipo de dados. Todos os acessos ao banco de dados de metadados são realizados por meio destas classes.

O diagrama de classes simplificado do módulo de planejamento e execução de experimentos (“Experiment Planner”) pode ser visto na Figura 13. Nele é possível visualizar as classes responsáveis por este módulo. Em especial a estrutura de dados “Scenario” que representa um

Figura 10 – Diagrama de Classes UML simplificado mostrando as classes do módulo Blob Storage

cenário de experimentos repleto de objetos do tipo “ScenarioOperation”, que representam as operações que serão realizadas no cenário. Ainda é possível visualizar a classe “ScenarioParser”, responsável por ler e escrever arquivos de cenário em XML e a classe “ScenarioExecutor”, responsável por executar os cenários desejados.

As classes que gerenciam e coordenam o sistema de dispersão em sua totalidade são

Figura 11 – Diagrama de Classes UML simplificado mostrando as classes que representam os tipos de dados do sistema

mostradas no diagrama da Figura 14. Nele, é possível observar a classe principal (“StoreSafeManager”) e as classes responsáveis por gerenciar os metadados (“DatabaseManager”) e o armazenamento (“StorageManager”).

As classes do módulo de seleção automática de estratégias de dispersão são apresentadas na Figura 15. É possível observar a interface “ISelector” e a sua primeira implementação “LPSelector”, que é detalhada nesta dissertação. A classe “Module” representa um módulo e seus atributos de seleção principais.

Por fim, a Figura 16 mostra o módulo “Logger” e suas classes principais. Em especial as classes “RTInputStream” e “RTOutputStream”, que são os monitores dos canais de transferência de dados intra-aplicação e externo (provedores de nuvem). Eles funcionam como filtros aos tradicionais mecanismos “InputStream” e “OutputStream” da linguagem Java adicionando maneiras de mensurar a vazão do canal.

Na Figura 17 são apresentados os casos de uso que o sistema deve suportar. Neste diagrama é possível observar previamente as dinâmicas de interação do sistema, em especial as funções de armazenamento, deleção e atualização de arquivos que necessitam de outros processos para serem concluídos.

Figura 12 – Diagrama de Classes UML simplificado mostrando as classes que gerenciam os metadados

Figura 13 – Diagrama de Classes UML simplificado mostrando as classes do módulo Experiment Planner

Figura 14 – Diagrama de Classes UML simplificado mostrando as classes que gerenciam e coordenam o sistema

Figura 16 – Diagrama de Classes UML simplificado mostrando as classes do módulo Logger

Figura 17 – Diagrama de Casos de Uso UML simplificado mostrando as operações do usuário

4.4 Seleção de estratégias de dispersão multinuvem de arquivos - MSSF

Juntamente com a arquitetura flexível para dispersar e recuperar arquivos em ambientes multinuvem, surgiu a necessidade de se simplificar a utilização da ferramenta para usuários leigos, o que motivou a pesquisa e o desenvolvimento de um mecanismo de seleção de armazenamento multinuvem. Esta seção aborda este mecanismo em detalhes, incluindo seus módulos, modelos e seu funcionamento.

4.4.1 Parâmetros de entrada do usuário

A interface do MSSF requer o seguinte conjunto de parâmetros de entrada do usuário:

- Caminho do arquivo;
- Número de falhas de provedores a suportar;
- Confidencialidade (não me preocupo, confidencial, muito confidencial);
- Tipo do arquivo (de acordo com as características do arquivo).

Figura 18 – Interface de usuário do MSSF

A Figura 18 mostra a interface de usuário do MSSF. O usuário necessita inserir apenas três parâmetros de QoS de entrada para cada arquivo. O primeiro é o número de falhas de provedores suportados, o segundo é o nível de confidencialidade desejado e o terceiro é o tipo de arquivo de acordo com os tipos de arquivos predefinidos. Este conjunto de parâmetros vai definir os parâmetros de otimização do modelo, tais como os pesos para cada parâmetro e as constantes a serem respeitadas.

Os níveis predefinidos de confidencialidade disponíveis ao usuário são: não me preocupo; confidencial; e muito confidencial. O nível “não me preocupo” significa que não há nenhum requisito especial em relação a confidencialidade do arquivo. O nível “confidencial” significa que o arquivo é confidencial e portanto requer um alto nível de confidencialidade, mas ainda mantendo um desempenho razoável. O nível “muito confidencial” almeja os melhores algoritmos e provedores em relação a confidencialidade para armazenar o arquivo, mesmo que eles tenham baixo desempenho.

O usuário irá escolher o tipo de arquivo de acordo com as características do arquivo representadas na Tabela 5. Com isso, o MSSF consegue aprimorar os parâmetros de otimização do modelo para se adequar às necessidades do usuário.

4.4.2 Modelo e motor de otimização

O módulo principal do MSSF é o seu motor de otimização, o qual é carregado com um modelo inteiro de otimização baseado no problema da mochila para selecionar a melhor combinação de módulos respeitando os requisitos e as necessidades do usuário.

A abordagem de otimização mista inteira foi escolhida pois o processo de dispersão é baseado em três categorias de módulos (dispersão, criptografia e compressão) e a quantidade de provedores, as quais são limitadas. Por exemplo, é improvável que o sistema tenha que escolher entre milhares de módulos e provedores, devido a característica de tais ambientes em nuvem reais.

O modelo é baseado no método de programação linear baseado na otimização global multi-critérios apresentada em [Zeng et al. \(2004\)](#) e compreende três elementos: a função objetivo, as variáveis de decisão e o conjunto de restrições. Este modelo permite maximizar um conjunto de parâmetros ao mesmo tempo que minimiza outros, o que é particularmente útil para o mecanismo.

O método escolhido para solucionar o problema de otimização utiliza programação linear inteira ([ZENG et al., 2004](#); [ARDAGNA](#); [PERNICI, 2007](#)). Este método requer variáveis de decisão que representem se o módulo especificado foi selecionado para determinada categoria. Ele também requer algumas regras de restrição para limitar o espaço de busca e direcionar o modelo de otimização. As equações que compõem este modelo de otimização estão especificadas abaixo:

Função objetivo:

Maximizar:

$$\left(\sum_{j=0}^3 \left(\frac{Q_j^{max} - Q_{k,j}}{Q_j^{max} - Q_j^{min}} * w_j \right) + \sum_{j=4}^7 \left(\frac{Q_{k,j} - Q_j^{min}}{Q_j^{max} - Q_j^{min}} * w_j \right) \right) \quad (4.1)$$

Sujeito a:

$$\sum_{i=0}^j w_i = 1, \quad w_i \in [0, 1] \quad (4.2)$$

$$\sum_{i=1}^{N_{ida}} D_i = 1, \quad D_i \in \{0, 1\} \quad (4.3)$$

$$\sum_{i=1}^{N_{enc}} E_i \leq 1, \quad E_i \in \{0, 1\} \quad (4.4)$$

$$\sum_{i=1}^{N_{comp}} C_i \leq 1, \quad C_i \in \{0, 1\} \quad (4.5)$$

$$Q_{k,j} \geq Q_j^{user} \quad (4.6)$$

$$\sum_{i=1}^{N_{prov}} P_i = P_{req}, \quad P_i \in \{0, 1\} \quad (4.7)$$

A equação 4.1 é a função objetivo do nosso modelo, e busca minimizar a sobrecarga de armazenamento (j_0), custo de armazenamento do provedor (j_1), custo de banda de entrada dos provedores (j_2), custo de banda de saída dos provedores (j_3) e maximizar a confidencialidade (j_4), desempenho (j_5), disponibilidade (j_6) e a durabilidade (j_7).

$Q_{k,j}$ é o valor agregado para o j_{th} atributo de QoS, oferecido pela k_{th} combinação de módulos e w_j é o peso normalizado para o j_{th} atributo de QoS. Q_j^{max} é o valor máximo agregado e Q_j^{min} é o valor mínimo agregado para o atributo de QoS j oferecido por todas as combinações de módulos.

A equação 4.2 define que a soma de todos os (w_i) parâmetros de peso normalizados são iguais a um para garantir que todos os pesos estão utilizando a mesma escala.

As equações 4.3, 4.4 e 4.5 são responsáveis por selecionar os módulos. As variáveis $N_{ida}, N_{enc}, N_{comp}$ representam a quantidade de módulos disponíveis para cada categoria. As variáveis de decisão D_i, E_i, C_i indicam se o módulo i será escolhido ($=1$) ou não ($=0$) para as categorias algoritmo de dispersão, criptografia e compressão, respectivamente. É importante notar que o algoritmo de dispersão (Equação 4.3) é necessário e será sempre selecionado. Já os módulos de criptografia e compressão são opcionais.

Os requisitos de QoS do usuário são satisfeitos utilizando a Equação 4.6. Novamente, a variável $Q_{k,j}$ é o valor agregado para o j_{th} atributo de QoS, oferecido pela k_{th} combinação de módulos. Q_j^{user} é o valor requerido pelo usuário para o atributo de QoS j . Isto garante que a solução satisfaz os requisitos necessários do usuário para os atributos de QoS desejados.

A seleção dos provedores é representada pela Equação 4.7. N_{prov} é o número de provedores disponíveis para se escolher. P_i é a variável de decisão utilizada para indicar se o provedor i será selecionado ($=1$) ou não ($=0$) e P_{req} é o número de provedores que o modelo precisa escolher para o processo de dispersão. A Tabela 7 resume as variáveis do modelo.

Tabela 7 – Variáveis do modelo de otimização

Nome	Tipo	Descrição	Fonte
$Q_{k,j}$	Inteira	Valor agregado para o j_{th} atributo de QoS, oferecido pela k_{th} combinação de módulos	Solver
Q_j^{max}	Inteira	Valor máximo agregado para o atributo de QoS j	Solver
Q_j^{min}	Inteira	Valor mínimo agregado para o atributo de QoS j	Solver
D_i, E_i, C_i	Binária	Variáveis de decisão que indicam se o módulo i será selecionado para cada categoria	Solver
P_i	Binária	Variável de decisão que indica se o provedor i será selecionado	Solver
$N_{ida}, N_{enc}, N_{comp}$	Inteira	Quantidade de módulos disponíveis em cada categoria	Banco de dados
N_{prov}	Inteira	Número de provedores	Banco de dados
P_{req}	Inteira	Quantidade de provedores a se escolher	Sistema especialista
w_j	Real	Peso normalizado para o j_{th} atributo de QoS	Sistema especialista
Q_j^{user}	Inteira	Valor requerido pelo usuário para o atributo de QoS j	Sistema especialista

É importante mencionar que o modelo de otimização foi validado utilizando casos menores e com o auxílio de especialistas em otimização da *University of Leicester*. Ainda, este modelo foi publicado na conferência IEEE Cloud 2015 e também foi avaliado por diversos especialistas da área de computação em nuvem.

4.4.3 Obtenção dos parâmetros dos provedores

É importante recuperar os parâmetros de QoS dos provedores para prover ao modelo de otimização os valores atualizados. Para obter estes dados em tempo real, foi criado um serviço *daemon* que recupera os dados de alguns provedores automaticamente. Ele popula o banco de dados com os respectivos valores para disponibilidade, desempenho de leitura, desempenho de escrita, custo de armazenamento e custo de banda. A disponibilidade é oferecida checando periodicamente se o serviço está disponível. O desempenho de leitura e escrita são calculados pela média da vazão (em KBps) de cada operação e os custos são obtidos diretamente das páginas dos provedores por meio de mensagem JSON. O custo de armazenamento é o custo para se armazenar um *gigabyte* de dados durante um mês. O custo de banda é o custo para se transferir um *gigabyte* de dados. A durabilidade é obtida do acordo de serviços dos provedores e

é mensurada em termos de novezes³. A Tabela 8 mostra um exemplo dos parâmetros de seleção recuperados dos provedores (de maneira anônima).

Tabela 8 – Provedores

Prov.	Disp.	Dur.	Desemp. leitura.	Desemp. escrita	Confid.	Custo armazen.	Custo banda entrada	Custo banda saída
P1	99.9	9	350	80	75	10	0.02	0.02
P2	99.99	11	653	115	75	15	0.06	0.06
P3	99	8	832	80	75	10	0.05	0.05
P4	99.999	12	221	35	75	7	0.01	0.01

4.4.4 Sistema especialista de ajuste dos pesos dos parâmetros de QoS baseado em regras

O primeiro peso de parâmetro de QoS é a confidencialidade, a qual é diretamente baseada no conjunto de entrada do usuário. Os outros pesos de parâmetros de QoS são definidos a partir de um sistema especialista composto de regras pré-definidas que consideram o conjunto de entrada do usuário, tamanho do arquivo e extensão do arquivo. As regras podem aumentar ou diminuir o valor de cada peso para refletir as características de dispersão desejadas.

Figura 19 – Fluxograma simplificado do sistema especialista baseado em regras

A Figura 19 mostra o fluxo do sistema especialista. Ela também apresenta como um exemplo (em caixas pontilhadas redondas do lado direito) um arquivo de filme (extensão .avi

³ <http://stackoverflow.com/questions/4179027/99-99-durability-what-does-it-mean>

com 850MB) escolhido para ser disperso. O sistema especialista do MSSF é capaz de reconhecer que o arquivo é um filme (devido a sua extensão e seu tamanho) e vai calcular (aumentando ou diminuindo) o peso dos parâmetros de QoS de acordo com a regra “Movie” existente. Algumas das regras criadas foram baseadas em um questionário com usuários ⁴, na literatura e na opinião de especialistas de armazenamento de dados.

As regras são armazenadas no banco de dados, porém para mostrar uma idéia do formato delas serão apresentadas as regras “Movie” (para filmes) e a genérica “BigFile” (para arquivos grandes) utilizando a notação de condição-ação abaixo:

```
Rule Movie:
if File.MIME=video && File.SIZE >= 600MB
then w_prf_rd=*1.25 and w_prf_writ=*0.6 and w_sto=*1.15
    and w_stcost=*1.15 and w_bw_in=*0.7 and w_dur=*1.15
```

```
Rule BigFile:
if File.SIZE >= 100MB
then w_perf_rd=*0.6 and w_avail=*1.2 and w_dur=*1.2
```

Como pode ser observado, as regras descrevem as mudanças que serão feitas nos pesos para os parâmetros de QoS baseados no tipo do arquivo; por exemplo para um filme que tende a ser mais lido do que escrito, o desempenho de escrita é mais importante.

4.4.5 Fluxo de trabalho

O MSSF é composto de três módulos principais: a interface gráfica de usuário (em inglês, GUI); um sistema especialista; e o módulo de otimização inteira linear. A Figura 20 mostra o fluxo de trabalho para o processo de seleção. Primeiramente, o usuário precisa prover os parâmetros de entrada para o arquivo desejado. Então, o sistema especialista utiliza algumas regras pré-definidas para influenciar no peso de cada um dos parâmetros de QoS. Finalmente, o MSSF executa o otimizador (*solver*) com os parâmetros de QoS calculados e seleciona a opção de dispersão ótima (quais provedores utilizar, o algoritmo de dispersão e os módulos) para dispersar o arquivo.

4.5 Metodologia

A metodologia para a avaliação do modelo e da hipótese foi composta por algumas fases complementares especificadas abaixo:

1. **Estudo de tecnologias para utilização no modelos:** Nesta etapa foram estudadas as tecnologias que serão utilizadas no projeto, incluindo os algoritmos, as formas de geração

⁴ <https://www.surveymonkey.com/results/SM-FML36X87/>

Figura 20 – Fluxo de trabalho do MSSF

da chave de acesso e localização do arquivo, canais de comunicação, linguagens de programação, tecnologias para disponibilização do serviço, estrutura física e lógica necessária para o protótipo e formas de avaliação do protótipo.

2. **Coleta de parâmetros e métricas:** Nesta etapa foram definidos, de acordo com o estado da arte, os parâmetros e métricas que devem ser coletados para ser possível obter comparações reais com outros trabalhos semelhantes, com o objetivo de avaliar a adequação e a eficiência do modelo proposto. As principais decisões tomadas no projeto favoreceram estes parâmetros e métricas detectados como importantes e necessários.
3. **Algoritmos utilizados e propostos:** Este trabalho avaliou diferentes algoritmos de dispersão de informações (*IDA*) buscando por características desejadas, como segurança e desempenho. Para a tarefa de seleção não existia nenhum algoritmo, por isso nesta fase um algoritmo foi proposto para realizar este trabalho.
4. **Modelagem:** Nesta etapa, o projeto foi modelado e formalizado para auxiliar na criação do protótipo. Para esta fase foram criados diversos diagramas da arquitetura e de funcionamento do modelo. Para esta fase foram utilizados conceitos de engenharia de software, além de abordagens para validar o modelo antes de prosseguir para a fase de prototipação.
5. **Prototipação e testes:** Nesta fase, foi criado um protótipo funcional a partir do modelo criado e validado anteriormente. Este protótipo foi testado, buscando verificar se ele atende aos requisitos funcionais e não-funcionais do modelo proposto.

6. **Planejamento de experimentos e coleta dos dados:** Esta etapa compreendeu o planejamento dos experimentos e de como foi feita a coleta de dados. Esta etapa mereceu uma atenção especial, pois todos os resultados e conclusões do projeto foram feitos a partir dos dados coletados nesta fase. Os experimentos devem abranger ao menos dois ambientes, um interno e controlado, em que seja possível determinar, obter e controlar todas as variáveis que influem na avaliação do protótipo. Este ambiente controlado foi utilizado para avaliar o MSSF. Já o segundo ambiente foi um ambiente real, em que se utilizaram serviços de nuvem existentes, como o *Box.com*, *Cubby*, *SME*, dentre outros.
7. **Análise dos resultados:** Esta foi a etapa final da metodologia, na qual todos os dados obtidos da fase anterior foram estudados e analisados levando em consideração fatores estatísticos, como a distribuição, intervalo de confiança, variância e erro.

A Figura 21 apresenta um fluxograma simplificado das etapas da metodologia de pesquisa.

Figura 21 – Fluxograma simplificado da metodologia

4.6 Ferramentas para o Desenvolvimento

1. Softwares

Para o desenvolvimento do projeto foram utilizadas diversas aplicações. Estas aplicações incluem softwares de *Benchmark* de serviços em nuvem, softwares estatísticos, otimizadores lineares, softwares para programação e depuração, dentre outros.

2. Infraestrutura

A infraestrutura disponível no parque computacional do *LaSDPC* é composta atualmente por 3 *clusters Beowulf* com capacidade de isolamento de máquinas do *cluster* de modo a conseguir obter uma arquitetura semelhante, embora de menor escala, às encontradas nos provedores atuais de serviços na nuvem. Também existe uma arquitetura distribuída em um dos *clusters* denominada *WSArch* (ESTRELLA; SANTANA; SANTANA, 2011), que já possui recursos de avaliação de desempenho de aplicações distribuídas, além de possibilidades de se realizar um teste extenso e com diversas variáveis a serem avaliadas.

3. Outros

O projeto utilizou algumas ferramentas para facilitar a comunicação e o gerenciamento de tarefas entre os envolvidos, possibilitando a gestão eficiente de recursos e prazos viabilizando a integração com outros trabalhos. Para possibilitar este controle, será utilizada a ferramenta *Projects-LaSDPC*⁵, desenvolvida por meio da plataforma Open Source Moodle (MOODLE.ORG, 2013). Nesta ferramenta é possível gerenciar prazos, reuniões, alocação de recursos e todos os documentos gerados durante o projeto. Por esta ferramenta ser disponibilizada por meio de uma plataforma *Web*, é possível acessá-la de qualquer lugar do mundo, viabilizando inclusive o trabalho remoto entre os participantes. Outra ferramenta a ser utilizada para gerenciamento dos materiais de leitura, como artigos e livros será a plataforma Mendeley® (HENNING; REICHEL, 2008). Esta plataforma disponibiliza o armazenamento destes materiais na nuvem, possibilitando o acesso aos artigos da área de qualquer computador conectado à *Internet*.

4.7 Considerações Finais

Nesta seção foi apresentado o trabalho de pesquisa, as contribuições que ele trará, algumas decisões de projeto do sistema, a metodologia de desenvolvimento, as ferramentas de desenvolvimento, publicações geradas e criadas. A próxima seção irá apresentar uma avaliação experimental do trabalho desenvolvido.

⁵ <http://projects.lasdpic.icmc.usp.br>

AVALIAÇÃO EXPERIMENTAL

5.1 Considerações Iniciais

Neste capítulo são mostradas as avaliações da FlexSky e do módulo de seleção MSSF. A FlexSky é avaliada por meio de um estudo de caso, com avaliações de desempenho e da facilidade de uso. Já o MSSF é avaliado por meio de uma análise de desempenho e por um teste de usabilidade.

5.2 Avaliação da FlexSky - Estudo de caso

Um estudo de caso baseando-se em um caso prático fictício de utilização da ferramenta FlexSky foi realizado. Ele foi baseado em uma competição para selecionar um estudante para uma bolsa relacionada ao projeto.

O objetivo deste estudo de caso fictício foi selecionar um estudante para uma bolsa de estudos disponível para este projeto. Este estudo de caso é composto de três tarefas para testar inicialmente a flexibilidade e a usabilidade da FlexSky.

5.2.1 Ambiente de testes

Os experimentos deste estudo de caso foram realizados utilizando um ambiente de testes único, no qual a plataforma FlexSky foi executada.

A máquina em que foram executados os testes possuía as seguintes configurações:

- **Processador:** Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz (2 Cores)
- **Memória:** 2048MB
- **Rede:** Realtek RTL8111/8168B

O sistema operacional utilizado tinha as seguintes configurações:

- **S.O.:** Ubuntu 12.04
- **Kernel:** 3.2.0-40-generic (x86-64)
- **Desktop:** Unity 5.18.0
- **Java:** Java(TM) SE Runtime Environment (build 1.7.0-15-b03)
- **Sistema de Arquivos:** ext4

O ambiente de teste também apresentava falhas de provedores, criadas aleatoriamente através da FlexSky para testar a adaptabilidade do sistema. Todas as variáveis foram mensuradas individualmente para cada módulo.

Para diminuir ao máximo a influência de fatores externos e não controlados, optou-se por fazer todo o processo de teste automático, utilizando a FlexSky em um ambiente controlado.

Este ambiente controlado foi obtido iniciando o sistema operacional em modo de texto e com todos os serviços desnecessários desativados. Ainda, visando evitar influências do cache, cada uma das replicações utilizou um arquivo distinto gerado automaticamente.

Os arquivos XML de cenário do experimento estão disponíveis no apêndice desta monografia.

É importante ressaltar que devido ao caráter de “disco virtual” dos provedores de armazenamento, suas características internas estão indisponíveis.

5.2.2 Metodologia

A primeira tarefa foi apresentar a FlexSky, juntamente com sua documentação para dois estudantes com conhecimento básico da linguagem Java e pedir para cada um deles desenvolver um módulo de um algoritmo de dispersão de informações (IDA) na arquitetura.

Os estudantes tinham que analisar a documentação disponibilizada e desenvolver um módulo IDA baseados no conhecimento prévio anterior. Foi permitido reutilizar bibliotecas existentes caso fosse desejado. O único requisito foi que o método deveria armazenar os arquivos nos provedores de nuvem com algum grau de redundância.

A segunda tarefa foi pedir para os estudantes construírem um módulo *pipeline* simples para a FlexSky. Este módulo precisava apenas codificar/decodificar arquivos utilizando o algoritmo *Pipe64* (FREED; BORENSTEIN, 1996) antes de armazenar o arquivo. Então, foi dado uma semana para cada candidato estudar e desenvolver os módulos pedidos.

Um dos candidatos criou um algoritmo de dispersão baseado no algoritmo de Rabin (RABIN, 1989b) utilizando uma biblioteca existente. Este módulo foi denominado de *Rabin*. O

outro estudante criou um módulo de replicação que copiava o arquivo para todos os provedores de nuvem. Este módulo foi denominado *Replicate*. Ambos candidatos desenvolveram um módulo de conversão *Pipe64* similar utilizando bibliotecas existentes do Java.

Para desenvolver os módulos, cada um deles precisou criar duas classes baseadas em classes abstratas da FlexSky. Foi então pedido a opinião dos candidatos na facilidade da FlexSky utilizando três questões objetivas variando de 1-“muito difícil” para 5-“muito fácil”. A Tabela 9 apresenta as questões, juntamente com suas respostas.

Pergunta	Usuário 1 (Rabin)	Usuário 2 (Replicate)
Quão fácil foi entender o código/documentação oferecido?	4	3
Quão fácil foi implantar os módulos requisitados na FlexSky?	5	4
Quão fácil foi entender os erros e testar/debugar os módulos utilizando a arquitetura FlexSky?	4	4

Tabela 9 – Perguntas de usabilidade e suas respostas

Após estas tarefas, foi necessário realizar uma avaliação justa dos módulos desenvolvidos utilizando ambientes em nuvem reais compostos de provedores de armazenamento existentes. Portanto, a terceira tarefa para finalizar o estudo de caso foi criar e executar em cenário de experimento com a FlexSky, utilizando o módulo planejador de experimentos para avaliar e comparar o desempenho dos módulos criados por cada estudante.

Para esta tarefa, foram definidos fatores e níveis utilizando um design de experimento fatorial completo. A Tabela 10 mostra estes fatores e níveis considerados. Os fatores fixados foram os seguintes:

- Número de replicações: 10;
- Número de provedores de armazenamento na nuvem **reais**: 10 (utilizados como discos virtuais de armazenamento);
- Nível de disponibilidade: tolerar pelo menos uma falha de um provedor;

As variáveis de resposta definidas foram: Tempo gasto para a dispersão do arquivo, tempo gasto para a recuperação do arquivo, vazão de dispersão do arquivo, vazão de recuperação do arquivo e a sobrecarga no espaço de armazenamento gerado para cada uma das configurações de experimentos. A sobrecarga é calculada dividindo o tamanho total somado das partes ao final do processo de dispersão pelo tamanho inicial do arquivo. Portanto, a fórmula para calcular a sobrecarga é $E_{final}/E_{inicial}$ no qual E é o espaço ocupado pelo arquivo ou pelas partes.

Essas variáveis permitem expressar a análise em termos de: (1) Tempo gasto para o armazenamento e recuperação da mensagem (e custo computacional real efetivo resultante da

Fatores variáveis	Níveis
Algoritmo de Dispersão	Rabin
	Replicate
Tamanho do arquivo	1Mb
	10Mb
File Pipeline	None
	Base64Pipe

Tabela 10 – Fatores e níveis variáveis utilizados no experimento.

escolha entre as configurações; e (2) Proporção de sobrecarga gerada e custo real subsequente de armazenamento.

5.2.3 Resultados

Figura 22 – Tempo das operações da FlexSky

A Figura 22 mostra os tempos e a Figura 23 mostra a vazão das operações e dos módulos de dispersão individualmente. Os tempos de *upload* e *download* são o tempo total do início até o envio de todas as partes. Os tempos do IDA são os tempos de CPU requeridos para o processo de dispersão e junção das partes. Estes dados tornam possível comparar o desempenho destas funcionalidades individualmente e combinados.

Figura 23 – Vazão das operações na FlexSky

Gráfico de superfície "Rabin IDA"

Figura 24 – Gráfico de superfície Rabin IDA

Gráfico de superfície "Replicate IDA"

Figura 25 – Gráfico de superfície Replicate IDA

A Figura 24 e a Figura 25 mostra o gráfico de superfície dos algoritmos “Rabin” e “Replicate”, respectivamente. O gráfico considera a vazão da dispersão (KBps), a vazão da junção (KBps) e a sobrecarga no tamanho. Para ambos os algoritmos é possível concluir que a vazão da dispersão é inversamente proporcional a sobrecarga no tamanho. Quanto maior o tamanho final, menor é a vazão de dispersão. Já a vazão na junção do arquivo é diretamente proporcional a sobrecarga no tamanho final. Isto significa que quanto maior o tamanho final, mais rápido é para recuperar o arquivo.

Utilizando ambas as figuras (Figuras 24 e 25) é possível observar que o algoritmo “Rabin” gera tamanhos menores de arquivo, porém sua recuperação é mais lenta comparando com o “Replicate”. Já o “Replicate”, por outro lado, é mais lento do que o “Rabin”. Comparando ambas as figuras, é possível concluir que o vencedor é o estudante que criou o algoritmo “Rabin”, o qual é mais complexo para desenvolver e melhor do que o “Replicate”.

A informação para se verificar o desempenho individual dos módulos do sistema também é uma medida muito útil. A Figura 26 mostra o desempenho do módulo *Pipe64* como um exemplo.

A FlexSky também consegue mensurar a operação de cada parte individualmente. A Figura 27 mostra a média da vazão para cada provedor de nuvem utilizado, e um resumo da sobrecarga de tamanho de arquivo para cada algoritmo de dispersão utilizando as configurações pré-definidas combinadas com o módulo *Pipe64*.

Figura 26 – Análise do módulo Pipe Base64

Figura 27 – Vazão dos provedores de nuvem e Sobrecarga de armazenamento

5.3 Avaliação do *Multi cloud Storage Selection Framework* (MSSF)

O módulo MSSF foi avaliado utilizando dois experimentos. O primeiro avaliou o desempenho do MSSF para verificar se ele é uma opção viável para ambientes de armazenamento em nuvens públicas. O segundo experimento é composto de um teste de usabilidade com usuários reais para verificar a facilidade de se utilizar o MSSF e se ele armazena os arquivos de acordo com os perfis de usuário e as propriedades de arquivo.

5.3.1 *Ambiente de testes*

Todos os experimentos foram realizados utilizando um ambiente de testes único, no qual a plataforma FlexSky foi executada.

A máquina em que foram executados os testes possuía as seguintes configurações:

- **Processador:** Intel Core i3-2330M 2,2 GHz (2 Cores)
- **Placa Mãe:** HP EP45-UD3L
- **Chipset:** Intel 4 DRAM + ICH10
- **Memória:** 4096MB
- **Disco:** 500GB Hitachi H87-0
- **Gráficos:** ATI Radeon 7670 256MB
- **Rede:** Realtek RTL8111/8168B

O sistema operacional utilizado tinha as seguintes configurações:

- **S.O.:** Windows 7 SP1
- **ILP Solver:** GLPK solver v4.55 ([MAKHORIN, 2008](#)).
- **Java:** Java(TM) SE Runtime Environment (build 1.7.0-15-b03)
- **Sistema de Arquivos:** NTFS

Para diminuir ao máximo a influência de fatores externos e não controlados, optou-se por fazer todo o processo de teste automático, utilizando a FlexSky em um ambiente controlado.

Este ambiente controlado foi obtido iniciando o sistema operacional em modo de texto e com todos os serviços desnecessários desativados. Os arquivos XML de cenário do experimento estão disponíveis no apêndice desta monografia.

5.3.2 Avaliação de desempenho

O primeiro conjunto de experimentos foi criado para avaliar o desempenho do MSSF e adequação a ambientes de nuvens reais com diversos servidores e módulos. O objetivo foi avaliar apenas o desempenho do mecanismo de seleção e não outros aspectos, como a comunicação.

Foi criado um experimento fatorial completo utilizando a plataforma FlexSky. Foi avaliada a variável de resposta tempo de seleção com dez replicações para cada experimento. O tempo de seleção representa o tempo necessário para o motor de otimização linear inteira descobrir a combinação de módulos e servidores que leva a solução ótima.

A Tabela 11 apresenta os fatores e níveis considerados. O número de módulos é a quantidade total de módulos (IDA, criptografia e compressão) carregados no sistema. Os provedores são os lugares possíveis para se armazenar as partes. O conjunto de parâmetros de entrada do processo de seleção são:

```
MIN_SEC=10;MIN_PERF=10;MIN_STO=10;
WEIGHT_SEC=0.4;WEIGHT_PERF_READ=0;
WEIGHT_PERF_WRITE=0.1;WEIGHT_STO=0;
WEIGHT_STOCOST=0.1;WEIGHT_BWCOST_IN=0.1;
WEIGHT_BWCOST_OUT=0.1;WEIGHT_AVAIL=0.2;
WEIGHT_DUR=0;PROV_REQ=75%.
```

Adicionalmente, um arquivo de configuração do cenário em XML para a FlexSky está disponível para este experimento (ele é fornecido no apêndice desta monografia).

Fatores	Níveis
Número de Módulos	300
	3000
	30000
Número de Provedores	100
	1000
	10000

Tabela 11 – MSSF - Fatores e níveis utilizados nos experimentos.

A Figura 28 mostra o tempo de seleção. O eixo x representa a combinação de fatores do experimento e o eixo y representa o tempo de seleção em milissegundos. É possível observar que mesmo com valores altos como dez mil provedores e trinta mil módulos, a média do tempo de seleção ficou abaixo de sete segundos, no cenário maior. No cenário de menor escala (cem provedores e trezentos módulos), mais provável para um usuário normal, a média do tempo de seleção foi de 180,6 milissegundos. Isto indica que o MSSF consegue selecionar provedores rapidamente, mesmo em cenários mais complexos.

Figura 28 – Tempo de seleção

A Figura 29 mostra a influência dos fatores no tempo de seleção. O tamanho da barra indica a influência de cada um dos fatores na variável de resposta. De acordo com esta análise estatística, o fator número de provedores apresenta uma influência ligeiramente superior no tempo de seleção do que o fator número de módulos.

5.3.3 Teste de Usabilidade

Um teste de usabilidade foi criado e executado para verificar a acurácia de acordo com as necessidades do usuário. Este teste foi conduzido com cinco usuários de diversos meios (acadêmico, computação, negócios e pessoal). Ele buscou verificar se os usuários iriam ser capazes de inserir os parâmetros de entrada de maneira correta de acordo com cada tipo de arquivo. Ele foi composto de quatro tarefas envolvendo quatro tipos de arquivos distintos apresentados abaixo:

1. Um arquivo de documento (.docx) de um relatório de negócios com dados financeiros confidenciais da empresa que o usuário trabalha.
2. Um manual de instruções público (.pdf) que o usuário necessita ler todos os dias no trabalho.

Figura 29 – Influência estatística dos fatores

3. Uma imagem (.jpg) da infância do usuário.
4. Um log de acesso de uma empresa que é automaticamente escrito.

Foi utilizado o método *think aloud* (LEWIS; RIEMAN, 1993) para o teste de usabilidade. Este método encoraja o usuário a falar sobre o que ele está pensando enquanto executa as ações. Isto é útil para verificar se o fluxo de pensamentos do usuário corresponde com o fluxo desejado para o sistema e verificar se os parâmetros de entrada são adequados para o usuário. Para este teste, foram utilizados cinco usuários, o que é suficiente para detectar a maior parte dos problemas de usabilidade (NIELSEN; LANDAUER, 1993).

As tarefas, juntamente com uma breve descrição dos tipos de arquivos e do método de teste de usabilidade *think aloud* foram apresentados para cada usuário. Posteriormente, foi mostrada a pasta em que os arquivos estão localizados e o teste iniciou. Ao final do teste, foi comparado o resultado da seleção de dispersão de acordo com um especialista em armazenamento. Isto tornou possível avaliar se os parâmetros de entrada são adequados as características do arquivo e descobrir erros de usabilidade na interface gráfica.

A Tabela 12 sumariza o resultado do teste de usabilidade. Todos os usuários foram capazes de completar todas as tarefas com sucesso. Os parâmetros de entrada foram ligeiramente diferentes, porém o sistema carregado com as regras pré-definidas foi capaz de ajustar o peso dos

Tabela 12 – Resultados do teste de usabilidade

Usuário/Tarefa	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Obs. de Usabilidade
Usuário 1	✓	✓	✓	✓	Confortável
Usuário 2	✓	✓	✓	✓	Pasta ao invés de seleção de arquivos; Confortável
Usuário 3	✓	✓	✓	✓	Confortável
Usuário 4	✓	✓	✓	✓	Pediu por uma interface avançada; Queria limitar por preço; Confortável
Usuário 5	✓	✓	✓	✓	Confortável

parâmetros, o que levou a uma estratégia de dispersão na nuvem adequada. Os usuários também se sentiram confortáveis com os parâmetros de entrada.

O usuário 2 sugeriu uma estratégia baseada em pastas ao invés de baseada em arquivos, pois para ele foi mais prático selecionar as propriedades para cada pasta e então armazenar os arquivos nas pastas desejadas. O usuário 4 (com conhecimento de computação) pediu por uma interface mais avançada na qual ele pudesse controlar todos os parâmetros diretamente. Ele também pediu por uma opção para limitar a escolha de acordo com o custo de armazenamento da estratégia selecionada.

Apesar de que o módulo leva em consideração o custo total, foi decidido não pedir este parâmetro ao usuário, pois eles exigem conhecimentos avançados. A maioria dos modelos de negócios dos provedores de armazenamento públicos são baseados no tamanho de armazenamento disponível e não na quantidade de dados armazenados, portanto em cenários reais não é prático para o usuário especificar o custo de armazenamento antes do envio de arquivos. Apesar da decisão de não pedir este parâmetro do usuário, o modelo suporta esta variável e uma interface de usuário avançada seria capaz de fornecer tais opções de custo ao usuário.

5.4 Considerações Finais

Esta seção apresentou uma avaliação da FlexSky e do módulo de seleção MSSF. Todo o experimento foi feito com base no planejamento de experimentos de avaliação de desempenho e seguiu uma metodologia rígida de execução e planejamento. Os resultados indicaram que os produtos avaliados foram consistentes e estão adequados para serem publicados e utilizados em ambientes reais.

CONCLUSÕES E TRABALHOS FUTUROS

6.1 Considerações iniciais

Esta seção conclui a dissertação, apresenta o cronograma e as etapas do projeto, que seguiu conforme planejado anteriormente. Ainda, esta seção apresenta possíveis trabalhos futuros e desdobramentos do presente trabalho.

6.2 Conclusão

O estudo de ferramentas e técnicas de armazenamento multinuvem representa avanços importantes na literatura de armazenamento em ambientes de nuvem, inclusive gerando novas perspectivas e desafios a serem superados.

Neste contexto, este projeto de mestrado obteve diversas contribuições para o estado da arte do armazenamento de arquivos em ambientes multinuvem utilizando técnicas de dispersão. Em especial, seus dois produtos finais (FlexSky e MSSF) contribuem para facilitar o estudo de novas técnicas e mecanismos neste contexto.

A FlexSky vai facilitar o estudo de novos algoritmos e módulos relacionados ao armazenamento de arquivos multinuvem, criando um ambiente flexível, heterogêneo e automatizado para se experimentar e testar novas funcionalidades. Isto é possível devido à sua arquitetura modular, na qual um pesquisador precisa desenvolver um módulo que pode ser acoplado diretamente a arquitetura. Isto possibilitará avaliar novas técnicas de dispersão, compressão, criptografia, mecanismos de segurança e avaliação de interfaces de usuário neste ambiente.

Já o módulo de seleção MSSF representa um avanço no estado da arte da seleção de estratégias de dispersão de arquivos em ambientes multinuvem. Ele representa uma etapa a partir da qual outros estudos podem ser criados com outras técnicas de seleção. Ao prover uma maneira de selecionar de maneira ótima a estratégia de dispersão, isto possibilita a comparação com

outras técnicas de seleção.

O projeto de mestrado foi executado com êxito e no prazo recomendado obtendo diversas contribuições.

6.3 Contribuições

Este trabalho apresenta duas contribuições relevantes para o estado da arte do armazenamento de arquivos em ambientes multinuvem, sendo estas:

- Criar uma arquitetura para dispersão de arquivos que possibilite (FlexSky):
 - Flexibilizar o nível de segurança;
 - Flexibilizar a localização dos metadados;
 - Utilizar provedores de armazenamento existentes;
 - Não precisar executar código nos provedores;
 - Flexibilizar o nível de armazenamento e o espaço utilizado;
 - Fazer todo o processo de dispersão/recuperação no cliente;
 - Adicionar novos módulos para testar novas funcionalidades;
 - Executar experimentos em lote sem supervisão;
 - Estudar mecanismos de seleção de estratégias de dispersão.
- Criar um mecanismo de seleção de estratégias de dispersão multinuvem que seja possível (MSSF):
 - Selecionar automaticamente uma estratégia de seleção;
 - Reduzir a quantidade de parâmetros provenientes do usuário;
 - Considerar diversos tipos de arquivos.

6.4 Integração com Outras Pesquisas

Este projeto apresenta possibilidades de integração evolutiva com diversas outras pesquisas, já que o sistema proposto utilizará uma arquitetura modular, em que cada módulo possui uma função no projeto. Devido a essa característica, pesquisas específicas das funções de cada módulo podem ser aplicadas neste modelo. E na realidade, este projeto já foi pensado buscando integrar algumas pesquisas do laboratório, como por exemplo, a pesquisa do mestrando Lucas Junqueira Adami ([ADAMI et al., 2015](#)). Nesse trabalho, foi criado um sistema de *anycast* ([PARTRIDGE; MILLIKEN, 1993](#)) na camada de aplicação; eficiente, facilmente escalável e que pode ser utilizado por qualquer tipo de sistema distribuído. A contribuição deste projeto na presente

proposta de mestrado será na disponibilização da lista de melhores servidores de armazenamento de arquivos na nuvem em escala global, por meio da rede mundial de computadores.

6.5 Trabalhos futuros

São esperados diversos trabalhos futuros a partir do ambiente de testes FlexSky. Diversas novas técnicas e estratégias podem ser facilmente testadas e utilizadas. Atualmente já existem dois projetos de iniciação científica que estão utilizando a FlexSky, um avaliando técnicas de criptografia negável no contexto de armazenamento multinuvem e outro avaliando modelos de auditoria para armazenamento multinuvem. Adicionar o suporte ao monitoramento na utilização de memória, aprimorar as interfaces gráficas, adicionar suporte a mais tecnologias de armazenamento e módulos de arquivos.

Também sugere-se interessante a criação de módulos de auditoria na ferramenta, visando o estudo destes no ambiente de armazenamento multinuvem.

Já para trabalhos futuros do MSSF, é interessante criar outros algoritmos de seleção multinuvem e comparar o desempenho e a acurácia destes com o algoritmo baseado em otimização linear criado neste projeto de mestrado.

Outra possibilidade é integrar as técnicas de avaliação de armazenamento na nuvem para que a ferramenta compare e avalie novas técnicas de maneira autônoma.

6.6 Etapas cumpridas

Esta pesquisa seguiu um plano de trabalho. Este plano contém as atividades desenvolvidas juntamente com todas as etapas e fases do projeto.

6.6.1 Cronograma

Este projeto foi composto de múltiplas atividades que foram divididas em várias etapas, as quais estão discriminadas a seguir:

1. **Cumprimento de créditos referente às disciplinas do mestrado:** Para realizar a defesa da dissertação é necessário cumprir um total de 48 créditos durante os dois semestres de 2013.
2. **Preparação para o exame de Proficiência em Inglês:** Essa etapa compreende o período de preparação para obter o certificado de proficiência em língua inglesa.
3. **Análise crítica dos trabalhos relacionados:** As atividades iniciais compreenderam a análise e comparação de algoritmos existentes de dispersão de informações, algoritmos de

criptografia. Também foram estudados e analisados as avaliações, e os modelos usados para avaliar este tipo de sistema.

4. **Preparação para o Exame de Qualificação:** Esta fase incluiu a preparação para o exame de qualificação realizado no início do primeiro semestre de 2014.
5. **Desenvolvimento da proposta:** Esta fase englobou todas as etapas de prototipação, testes e avaliação do projeto. Esta foi uma das principais fases do projeto, em que ele foi desenvolvido. Esta fase demandou bastante esforço na proposta de algoritmos, modelagem, prototipação, teste e realização de experimentos para validar o modelo.
6. **Elaboração de artigos científicos:** Preparação de artigos que promovam a divulgação dos resultados alcançados e a apresentação das contribuições científicas.
7. **Estágio de pesquisa no exterior:** O candidato participou de um estágio no exterior pelo período de seis meses (bolsa BEPE) com o pesquisador Stephan Reiff-Marganiec na Universidade de Leicester, Inglaterra ([REIFF-MARGANIEC, 2013](#)). Durante este período foi desenvolvido o módulo de seleção MSSF.
8. **Redação da dissertação:** Fase em que ocorreu a redação da dissertação de mestrado, que foi entregue em Agosto de 2015, para a posterior defesa.

Na Tabela 13, estas atividades estão ilustradas de acordo com o tempo em que foram realizadas. É importante mencionar que o cronograma foi seguido conforme o planejamento inicial. Alguns artigos já foram publicados em conferências de alto impacto e outros já foram aprovados ou estão em fase final de submissão.

Fases	Período (por bimestres)													
	2013					2014						2015		
	2	3	4	5	6	1	2	3	4	5	6	1	2	3
1	✓	✓	✓	✓	✓									
2	✓	✓	✓	✓										
3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
4		✓	✓	✓	✓									
5		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
6						✓	✓	✓	✓	✓	✓	✓	✓	✓
7							✓	✓	✓					
8						✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabela 13 – Organização das atividades

6.7 Considerações finais

Esta seção apresentou o cronograma e as etapas que ainda faltam serem desenvolvidas, incluindo a possível integração com outras pesquisas do laboratório. Este projeto já rendeu

algumas publicações e artigos prontos para submissão. O projeto seguiu o cronograma calculado previamente e possui possíveis trabalhos futuros interessantes, principalmente versando em relação a melhoria da arquitetura e ao estudo de novas técnicas de seleção multinuvel.

PUBLICAÇÕES E SUBMISSÕES

Diante da possibilidade real de aplicação, o projeto de pesquisa resultou em contribuições: apresentação em conferência, submissões para olimpíadas de inovação e um artigo para ser submetido em uma revista de alto impacto.

- LIBARDI, R. M. ; ESTRELLA, J. C. ; NAKAMURA, L. H. V. ; NUNES, L. H. . Private Storage Based on Public Cloud Storage. BIN@Brazil (Business and Innovation Network) 2013. (Apresentação de Trabalho/Conferência ou palestra).
- LIBARDI, R. M. ; ESTRELLA, J. StoreSafe.it. Olimpíada USP de Inovação 2013. Categoria 1. (Apresentação de Trabalho/Conferência ou palestra).
- LIBARDI, R.M.; NAVES, M.V. B.; REIFF-MARGANIEC, S.; ESTRELLA, J. C., “MSSF: A Step towards User-Friendly Multi-cloud Data Dispersal,” Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on , vol., no., pp.952,953, June 27 2014-July 2 2014 doi: 10.1109/CLOUD.2014.138 (Artigo em conferência/Qualis B2)
- LIBARDI, R. M. ; REIFF-MARGANIEC, S.; NUNES, L. H.; ADAMI, L. J.; FERREIRA, C. H. G.; ESTRELLA, J. C. “MSSF: User-friendly multi-cloud data dispersal” Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on, June 27 2014-July 2 2014. (Artigo em Conferência/Qualis B2/Aceito para publicação - Taxa de aceitação: 15%).
- LIBARDI, R. M. ; ADAMI, L. J.; NUNES, L. H.; NAKAMURA, L. H. V.; REIFF-MARGANIEC, S.; FERREIRA, C. H. G.; ESTRELLA, J. C. “FlexSky: a Flexible Test Bed to Evaluate Distributed Multi-cloud Storage Studies”. (Artigo submetido/Journal PlosOne)

REFERÊNCIAS

ABU-LIBDEH, H.; PRINCEHOUSE, L.; WEATHERSPOON, H. Racs: a case for cloud storage diversity. In: **Proceedings of the 1st ACM symposium on Cloud computing**. New York, NY, USA: ACM, 2010. (SoCC '10), p. 229–240. ISBN 978-1-4503-0036-0. Disponível em: <http://doi.acm.org/10.1145/1807128.1807165>. Citado 4 vezes nas páginas 39, 40, 48 e 51.

ADAMI, L.; ESTRELLA, J.; REIFF-MARGANIEC, S.; LIBARDI, R. D. O. Quality of service on services selection using anycast techniques: A convergence analysis. In: **Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on**. [S.l.: s.n.], 2015. p. 806–809. Citado na página 96.

ALABOOL, H.; MAHMOOD, A. Review on cloud service evaluation and selection methods. In: **Research and Innovation in Information Systems (ICRIIS), 2013 International Conference on**. Washington, DC, USA: IEEE Computer Society, 2013. p. 61–66. Citado 3 vezes nas páginas 17, 40 e 41.

ARDAGNA, D.; PERNICI, B. Adaptive service composition in flexible processes. **IEEE Trans. Softw. Eng.**, IEEE Press, Piscataway, NJ, USA, v. 33, n. 6, p. 369–384, jun. 2007. ISSN 0098-5589. Disponível em: <http://dx.doi.org/10.1109/TSE.2007.1011>. Citado na página 74.

ARIAS, D.; RIVERA, W. Using grid computing to enable distributed radar data retrieval and processing. In: **Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06. Seventh International Conference on**. Washington, DC, USA: IEEE Computer Society, 2006. p. 321–326. Citado na página 37.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. **Commun. ACM**, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, apr 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>. Citado na página 23.

ASHRAF, K.; ANANE, R.; BORDBAR, B. File management in a mobile dht-based p2p environment. In: **Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on**. Piscataway, NJ, USA: IEEE Press, 2012. p. 415–422. ISSN 1550-445X. Citado na página 38.

AUGUSTINE, J.; MOLL, A. R.; MORSY, E.; PANDURANGAN, G.; ROBINSON, P.; UPFAL, E. Storage and Search in Dynamic Peer-to-Peer Networks. **ArXiv e-prints**, maio 2013. Citado na página 38.

BAUER, E.; ADAMS, R. Analyzing cloud reliability and availability. In: BAUER, E.; ADAMS, R. (Ed.). **Reliability and Availability of Cloud Computing**. John Wiley and Sons, Inc., 2012. p. 63–89. ISBN 9781118393994. Disponível em: <http://dx.doi.org/10.1002/9781118393994.ch4>. Citado na página 39.

BESSANI, A.; CORREIA, M.; QUARESMA, B.; ANDRÉ, F.; SOUSA, P. Depsky: dependable and secure storage in a cloud-of-clouds. In: **Proceedings of the sixth conference on Computer**

systems. New York, NY, USA: ACM, 2011. (EuroSys '11), p. 31–46. ISBN 978-1-4503-0634-8. Disponível em: <http://doi.acm.org/10.1145/1966445.1966449>. Citado 3 vezes nas páginas 39, 48 e 51.

BESSANI, A.; MENDES, R.; OLIVEIRA, T.; NEVES, N.; CORREIA, M.; PASIN, M.; VERISSIMO, P. Scfs: A shared cloud-backed file system. In: **2014 USENIX Annual Technical Conference (USENIX ATC 14)**. Philadelphia, PA: USENIX Association, 2014. p. 169–180. ISBN 978-1-931971-10-2. Disponível em: <https://www.usenix.org/conference/atc14/technical-sessions/presentation/bessani>. Citado 2 vezes nas páginas 49 e 51.

BIAN, J.; SEKER, R. The jigsaw secure distributed file system. **Computers and Electrical Engineering**, v. 39, n. 4, p. 1142 – 1152, 2013. ISSN 0045-7906. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045790613000244>. Citado na página 38.

BOWERS, K. D.; JUELS, A.; OPREA, A. Hail: a high-availability and integrity layer for cloud storage. In: **Proceedings of the 16th ACM conference on Computer and communications security**. New York, NY, USA: ACM, 2009. (CCS '09), p. 187–198. ISBN 978-1-60558-894-0. Disponível em: <http://doi.acm.org/10.1145/1653662.1653686>. Citado 2 vezes nas páginas 47 e 51.

CALDER, A. **Implementing Information Security Based on ISO 27001/ISO 27002: A Management Guide**. Van Haren Publishing, 2009. ISBN 9789087535414. Disponível em: <http://books.google.com.br/books?id=x-O4f1lrq8QC>. Citado na página 39.

CASAS, P.; FISCHER, H.; SUETTE, S.; SCHATZ, R. A first look at quality of experience in personal cloud storage services. In: **Communications Workshops (ICC), 2013 IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2013. p. 733–737. Citado na página 40.

CHEN, P.; PATTERSON, D. Storage performance-metrics and benchmarks. **Proceedings of the IEEE**, IEEE Press, Piscataway, NJ, USA, v. 81, n. 8, p. 1151–1165, Aug 1993. ISSN 0018-9219. Citado na página 40.

CHEN, P. M.; LEE, E. K.; GIBSON, G. A.; KATZ, R. H.; PATTERSON, D. A. Raid: high-performance, reliable secondary storage. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 26, n. 2, p. 145–185, jun. 1994. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/176979.176981>. Citado 2 vezes nas páginas 47 e 48.

CHEN, S.; LIU, J. Statistical applications of the poisson-binomial and conditional bernoulli distributions. **Statistica Sinica**, v. 7, n. 4, p. 875–892, 1997. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0031313967&partnerID=40&md5=3316a763aea912beb33d5bf32d2bd96b>. Citado na página 43.

DAEMEN, J.; RIJMEN, V. The block cipher rijndael. In: QUISQUATER, J.-J.; SCHNEIER, B. (Ed.). **Smart Card Research and Applications**. Springer Berlin Heidelberg, 2000, (Lecture Notes in Computer Science, v. 1820). p. 277–284. ISBN 978-3-540-67923-3. Disponível em: http://dx.doi.org/10.1007/10721064_26. Citado na página 37.

DINGLEDINE, R.; FREEDMAN, M. J.; MOLNAR, D. The free haven project: Distributed anonymous storage service. In: **In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability**. Berkeley, CA, USA: International Computer Science Institute (ICSI), 2000. p. 67–95. Citado na página 38.

ESTRELLA, J. C.; SANTANA, R. H. C.; SANTANA, M. J. **WSARCH: An Architecture for Web Services Provisioning with QoS Support - Performance Challenges**. Saarbrücken, DE: Saarbrücken : VDM Verlag Dr. Müller GmbH & Co, 2011. Citado na página 81.

FREED, N.; BORENSTEIN, N. **Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies**. United States: RFC Editor, 1996. Citado na página 84.

GAO, X.; SHAH, P.; YOGA, A.; KODGIRE, A.; NI, X. A survey of cloud storage systems. In: **Abstracts of the CloudCom 2010**. Indianapolis, IN, EUA: CloudCom, 2010. Citado 3 vezes nas páginas 17, 31 e 32.

GARG, S.; VERSTEEG, S.; BUYYA, R. Smicloud: A framework for comparing and ranking cloud services. In: **Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2011. p. 210–218. Citado na página 43.

GENS, F. **IT Cloud Services User Survey, pt.2: Top Benefits and Challenges**. 2008. <<http://hadoop.apache.org>>. (Acessado em 14 de março de 2013). Citado 2 vezes nas páginas 23 e 24.

G.NALINIPRIYA; KUMAR, R. Secure massive data storage with consistency and route control on the cloud. **IOSR Journal of Computer Engineering**, IOSR Journals, New York, NY, USA, v. 9, n. 1, p. 18–25, jan 2013. ISSN 2278-8727. Disponível em: <<http://www.iosrjournals.org/iosr-jce/papers/Vol9-Issue1/D0911825.pdf>>. Citado na página 24.

GOOGLE©. **Google Docs®**. 2013. <<https://docs.google.com>>. (Acessado em 14 de março de 2013). Citado na página 23.

_____. **Google Drive®**. 2013. <<https://drive.google.com>>. (Acessado em 14 de março de 2013). Citado na página 23.

HEINIG, G. Inversion of generalized cauchy matrices and other classes of structured matrices. In: BOJANCZYK, A.; CYBENKO, G. (Ed.). **Linear Algebra for Signal Processing**. Springer New York, 1995, (The IMA Volumes in Mathematics and its Applications, v. 69). p. 63–81. ISBN 978-1-4612-8703-2. Disponível em: <http://dx.doi.org/10.1007/978-1-4612-4228-4_5>. Citado na página 35.

HENNING, V.; REICHEL, J. Mendeley-a last. fm for research? In: IEEE. **eScience'08. IEEE Fourth International Conference on**. Piscataway, NJ, USA: IEEE Press, 2008. p. 327–328. Citado na página 81.

HUANG, H. H.; GRIMSHAW, A. S. Design, implementation and evaluation of a virtual storage system. **Concurr. Comput. : Pract. Exper.**, John Wiley and Sons Ltd., Chichester, UK, v. 23, n. 4, p. 311–331, mar 2011. ISSN 1532-0626. Disponível em: <<http://dx.doi.org/10.1002/cpe.1644>>. Citado na página 38.

HUME, A.; AL-HAZMI, Y.; BELTER, B.; CAMPOWSKY, K.; CARRIL, L.; CARROZZO, G.; ENGEN, V.; GARCIA-PEREZ, D.; PONSATI, J. J.; KUBERT, R.; LIANG, Y.; ROHR, C.; SEGHBROECK, G. V. Bonfire: A multi-cloud test facility for internet of services experimentation. In: KORAKIS, T.; ZINK, M.; OTT, M. (Ed.). **Testbeds and Research Infrastructure. Development of Networks and Communities**. Springer Berlin Heidelberg, 2012, (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, v. 44). p. 81–96. ISBN 978-3-642-35575-2. Disponível em: <http://dx.doi.org/10.1007/978-3-642-35576-9_11>. Citado 2 vezes nas páginas 50 e 51.

ITANI, W.; KAYSSI, A.; CHEHAB, A. Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures. **Dependable, Autonomic and Secure Computing, IEEE International Symposium on**, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 711–716, 2009. Citado 2 vezes nas páginas 31 e 32.

JACOBSON, N. **Basic Algebra I**. Dover Publications, Incorporated, 2009. ISBN 9780486471891. Disponível em: <<http://books.google.com.br/books?id=8M48mAEACAAJ>>. Citado na página 36.

JAMMALAMADAKA, R. C.; GAMBONI, R.; MEHROTRA, S.; SEAMONS, K.; VENKATASUBRAMANIAN, N. A middleware approach for outsourcing data securely. **Computers & Security**, v. 32, n. 0, p. 252 – 266, 2013. ISSN 0167-4048. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S016740481200106X>>. Citado 3 vezes nas páginas 39, 48 e 51.

KUMAR, A.; LEE, B. G.; LEE, H.; KUMARI, A. Secure storage and access of data in cloud computing. In: **ICT Convergence (ICTC), 2012 International Conference on**. Piscataway, NJ, USA: IEEE Press, 2012. p. 336–339. Citado 2 vezes nas páginas 15 e 32.

LE, S.; DONG, H.; HUSSAIN, F.; HUSSAIN, O.; MA, J.; ZHANG, Y. Multicriteria decision making with fuzziness and criteria interdependence in cloud service selection. In: **Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2014. p. 1929–1936. Citado na página 41.

LEFEBVRE, G.; FEELEY, M. J. Separating durability and availability in self-managed storage. In: **Proceedings of the 11th Workshop on ACM SIGOPS European Workshop**. New York, NY, USA: ACM, 2004. (EW 11). Disponível em: <<http://doi.acm.org/10.1145/1133572.1133576>>. Citado na página 42.

LEWIS, C.; RIEMAN, J. **Task-centered User Interface Design: A Practical Introduction**. University of Colorado, Boulder, Department of Computer Science, 1993. Disponível em: <<http://books.google.com.br/books?id=j8mOYgEACAAJ>>. Citado na página 93.

LI, M. On the confidentiality of information dispersal algorithms and their erasure codes. **CoRR**, abs/1206.4123, 2012. Disponível em: <<http://arxiv.org/abs/1206.4123>>. Citado na página 44.

LU, W.; ZHOU, Y.; LI, J.; YAN, B. An efficient parallel storage scheme in data grid environment. In: **Future Computer and Communication (ICFCC), 2010 2nd International Conference on**. Piscataway, NJ, USA: IEEE Press, 2010. v. 3, p. V3–595–V3–599. Citado na página 37.

LUNA, J.; FLOURIS, M.; MARAZAKIS, M.; BILAS, A. Providing security to the desktop data grid. In: **Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on**. Piscataway, NJ, USA: IEEE Press, 2008. p. 1–8. ISSN 1530-2075. Citado na página 37.

MAKHORIN, A. **GLPK (GNU linear programming kit)**. 2008. Citado na página 90.

MALKHI, D.; REITER, M. Byzantine quorum systems. **Distributed Computing**, Springer-Verlag, v. 11, n. 4, p. 203–213, 1998. ISSN 0178-2770. Disponível em: <<http://dx.doi.org/10.1007/s004460050050>>. Citado na página 48.

MAR, K. K. Secured virtual diffused file system for the cloud. In: **Internet Technology and Secured Transactions (ICITST), 2011 International Conference for**. Piscataway, NJ, USA: IEEE Press, 2011. p. 116–121. Citado 4 vezes nas páginas 25, 39, 48 e 51.

MESNIER, M.; THERESKA, E.; GANGER, G.; ELLARD, D.; SELTZER, M. File classification in self-* storage systems. In: **Autonomic Computing, 2004. Proceedings. International Conference on**. Piscataway, NJ, USA: IEEE Press, 2004. p. 44–51. Citado na página 45.

MILLER, F.; VANDOME, A.; JOHN, M. **Federal Standard 1037C**. VDM Publishing, 2010. ISBN 9786131799709. Disponível em: <<http://books.google.co.uk/books?id=SyEaYAAACAAJ>>. Citado na página 42.

MOODLE.ORG. **Moodle Sites**. 2013. "http://moodle.org/sites/". (Acessado em 14 de março de 2013). Citado na página 81.

NIELSEN, J.; LANDAUER, T. K. A mathematical model of the finding of usability problems. In: **Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems**. New York, NY, USA: ACM, 1993. (CHI '93), p. 206–213. ISBN 0-89791-575-5. Disponível em: <<http://doi.acm.org/10.1145/169059.169166>>. Citado na página 93.

PAPAIIOANNOU, T. G.; BONVIN, N.; ABERER, K. Scalia: An adaptive scheme for efficient multi-cloud storage. In: **Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis**. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012. (SC '12), p. 20:1–20:10. ISBN 978-1-4673-0804-5. Disponível em: <<http://dl.acm.org/citation.cfm?id=2388996.2389024>>. Citado na página 52.

PARTRIDGE, T. M. C.; MILLIKEN, W. **Host Anycasting Service**. 1993. RFC1546. Citado na página 96.

PETCU, D. Consuming resources and services from multiple clouds. **Journal of Grid Computing**, Springer Netherlands, v. 12, n. 2, p. 321–345, 2014. ISSN 1570-7873. Disponível em: <<http://dx.doi.org/10.1007/s10723-013-9290-3>>. Citado na página 52.

RABIN, M. O. Efficient dispersal of information for security, load balancing, and fault tolerance. **J. ACM**, ACM, New York, NY, USA, v. 36, n. 2, p. 335–348, abr 1989. ISSN 0004-5411. Disponível em: <<http://doi.acm.org/10.1145/62044.62050>>. Citado 3 vezes nas páginas 15, 33 e 35.

_____. _____. **J. ACM**, ACM, New York, NY, USA, v. 36, n. 2, p. 335–348, abr. 1989. ISSN 0004-5411. Disponível em: <<http://doi.acm.org/10.1145/62044.62050>>. Citado na página 84.

_____. Sequences. In: CAPOCELLI, R. M. (Ed.). New York, NY, USA: Springer-Verlag New York, Inc., 1990. cap. The information dispersal algorithm and its applications, p. 406–419. ISBN 0-387-97186-6. Disponível em: <<http://dl.acm.org/citation.cfm?id=93933.94096>>. Citado 3 vezes nas páginas 15, 33 e 34.

REHMAN, Z.; HUSSAIN, F.; HUSSAIN, O. Towards multi-criteria cloud service selection. In: **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on**. Piscataway, NJ, USA: IEEE Press, 2011. p. 44–48. Citado na página 41.

REHMAN, Z.; HUSSAIN, O.; HUSSAIN, F. IaaS cloud selection using mcdm methods. In: **e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on**. Piscataway, NJ, USA: IEEE Press, 2012. p. 246–251. Citado na página 41.

REHMAN, Z. ur; HUSSAIN, O.; HUSSAIN, F. Multi-criteria IaaS service selection based on QoS history. In: **Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on**. Piscataway, NJ, USA: IEEE Press, 2013. p. 1129–1135. ISSN 1550-445X. Citado na página 41.

REIFF-MARGANIEC, S. **Homepage for Stephan Reiff-Marganiec**. 2013. [Http://www.cs.le.ac.uk/people/srm13/](http://www.cs.le.ac.uk/people/srm13/). (Acessado em 14 de março de 2013). Citado na página 98.

RISTOV, S.; GUSEV, M.; KOSTOSKA, M. A new methodology for security evaluation in cloud computing. In: **MIPRO, 2012 Proceedings of the 35th International Convention**. Piscataway, NJ, USA: IEEE Press, 2012. p. 1484–1489. Citado na página 39.

RUIZ-ALVAREZ, A.; HUMPHREY, M. An automated approach to cloud storage service selection. In: **Proceedings of the 2Nd International Workshop on Scientific Cloud Computing**. New York, NY, USA: ACM, 2011. (ScienceCloud '11), p. 39–48. ISBN 978-1-4503-0699-7. Disponível em: <http://doi.acm.org/10.1145/1996109.1996117>. Citado na página 52.

_____. A model and decision procedure for data storage in cloud computing. In: **Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on**. New York, NY, USA: IEEE, 2012. p. 572–579. Citado na página 52.

SANTIS, A. D.; MASUCCI, B. On information dispersal algorithms. In: **Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on**. Piscataway, NJ, USA: IEEE Press, 2002. p. 410–. Citado 2 vezes nas páginas 32 e 34.

SCHNJAKIN, M.; KORSCH, D.; SCHOENBERG, M.; MEINEL, C. Implementation of a secure and reliable storage above the untrusted clouds. In: **Computer Science Education (ICCSE), 2013 8th International Conference on**. Piscataway, NJ, USA: IEEE Press, 2013. p. 347–353. Citado 3 vezes nas páginas 39, 49 e 51.

SPILLNER, J.; MULLER, J.; SCHILL, A. Creating optimal cloud storage systems. **Future Gener. Comput. Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 29, n. 4, p. 1062–1072, jun. 2013. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2012.06.004>. Citado 2 vezes nas páginas 49 e 51.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 3rd. ed. New York, NY, USA: Pearson Education, 2002. ISBN 0130914290. Citado 4 vezes nas páginas 15, 23, 29 e 30.

WARD, R. Osi network security and the ntc. In: BERTSON, T.; BETH, T. (Ed.). **Local Area Network Security**. Springer Berlin Heidelberg, 1989, (Lecture Notes in Computer Science, v. 396). p. 67–74. ISBN 978-3-540-51754-2. Disponível em: http://dx.doi.org/10.1007/3-540-51754-5_31. Citado na página 29.

WICKER, S. B.; BHARGAVA, V. K. (Ed.). **Reed-Solomon Codes and Their Applications**. New York, NY, USA: John Wiley & Sons, Inc., 1999. ISBN 0780353919. Citado na página 34.

XIAO, L.; YE, Y.; YEN, I.-L.; BASTANI, F. Evaluation and comparisons of dependable distributed storage designs for clouds. In: **High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on**. Piscataway, NJ, USA: IEEE Press, 2010. p. 152–161. ISSN 1530-2059. Citado na página 38.

YU, J.; LU, P.; ZHU, Y.; XUE, G.; LI, M. Towards secure multi-keyword top-k retrieval over encrypted cloud data. **IEEE Transactions on Dependable and Secure Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 99, n. PrePrints, p. 1, 2013. ISSN 1545-5971. Citado na página 24.

ZENG, L.; BENATALLAH, B.; NGU, A.; DUMAS, M.; KALAGNANAM, J.; CHANG, H. Qos-aware middleware for web services composition. **Software Engineering, IEEE Transactions on**, IEEE Press, Piscataway, NJ, USA, v. 30, n. 5, p. 311–327, May 2004. ISSN 0098-5589. Citado 2 vezes nas páginas 52 e 74.

ZHANG, Q.; LUO, B.; SHI, W.; ALMOHARIB, A. M. **CloudSafe: Storing Your Digital Asset in the Cloud-based Safe**. 2013. <<http://www.cs.wayne.edu/~weisong/papers/zhang13-cloudsafe.pdf>>. (Acessado em 14 de julho de 2013). Citado 3 vezes nas páginas 39, 49 e 51.

ZHAO, D.; BURLINGAME, K.; DEBAINS, C.; ALVAREZ-TABIO, P.; RAICU, I. Towards high-performance and cost-effective distributed storage systems with information dispersal algorithms. In: **Cluster Computing (CLUSTER), 2013 IEEE International Conference on**. Piscataway, NJ, USA: IEEE Press, 2013. p. 1–5. Citado na página 43.

ZHOU, M.; ZHANG, R.; XIE, W.; QIAN, W.; ZHOU, A. Security and privacy in cloud computing: A survey. In: **Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on**. Piscataway, NJ, USA: IEEE Press, 2010. p. 105–112. Citado na página 31.