Pós-Graduação em Ciência da Computação

"Uma Abordagem para Indexação e Buscas *Full-Text* Baseadas em Conteúdo em Sistemas de Armazenamento em Nuvem"

Por

Marco André Santos Machado

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife, Agosto/2013

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Marco André Santos Machado

"Uma Abordagem para Indexação e Buscas *Full-Text* Baseadas em Conteúdo em Sistemas de Armazenamento em Nuvem"

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Vinicius Cardoso Garcia*
Co-Orientador: *Frederico Araújo Durão*

Recife, Agosto/2013

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Machado, Marco André Santos
Uma abordagem para indexação e buscas Full-Text baseadas em conteúdo em sistemas de armazenamento em nuvem / Marco André Santos Machado. - Recife: O Autor, 2013.
xi, 61 f., il., fig., tab.

Orientador: Vinícius Cardoso Garcia.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui referências.

1. Engenharia de software. 2. Recuperação de informação. 3. Computação em nuvem. 4. Sistemas distribuídos. I. Garcia, Vinícius Cardoso (orientador). II. Título.

005.1 CDD (23. ed.) MEI2014 – 006

Dissertação de Mestrado apresentada por **Marco André Santos Machado** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Uma Abordagem para Indexação e Buscas Full-Text Baseadas em Conteúdo em Sistemas de Armazenamento em Nuvem**" orientada pelo **Prof. Vinicius Cardoso Garcia** e aprovada pela Banca Examinadora formada pelos professores:

_____

Profa. Bernadette Farias Lóscio
Centro de Informática / UFPE

_____

Prof. Rodrigo Elia Assad
Departamento de Estatística e Informática/ UFRPE

_____

Prof. Vinicius Cardoso Garcia
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 2 de setembro de 2013

_____

Prof. Edna Natividade da Silva Barros
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Eu dedico esta dissertação à meus pais por todo o incentivo
e apoio que sempre me deram em minha vida.

Agradecimentos

Primeiramente gostaria de agradecer a toda minha família, em especial à meus pais, André e Eliete, e irmãos, Mateus e Jéssica, por todo o amor, carinho e incentivo dado durante toda minha vida.

Agradeço a Felícia, minha namorada, por seu amor, amizade e apoio incondicional durante todos esses anos e por sempre me incentivar a seguir adiante.

Agradeço a Vinicius Garcia e Frederico Durão, respectivamente, meu orientador e co-orientador, por dedicarem tempo para me orientar durante o desenvolvimento deste trabalho e pelos seus conselhos, ensinamentos e cobranças.

Aos meus amigos da Ustore, em especial à Rodrigo Assad, Anderson Fonseca, Francisco Soares, Paulo Fernando, por toda a estrutura técnica disponibilizada e pelas inúmeras discussões relacionadas a este trabalho.

Aos amigos que fiz dentro do CIn, especialmente à Paulo Fernando, Thiago Vieira, Lenin Abadie, Francisco Airton, Dhiego Abrantes, Adriano Tito, Rodolfo Arruda, Jamilson Batista, João Emanoel, Bruno Felipe, Hélio Rodrigues, Kellyton Brito, Leandro Marques.

Agradeço a Bernadette Lóscio e Rodrigo Assad por aceitarem fazer parte da banca da minha defesa e pelas críticas e conselhos dados, que possibilitaram melhorar este trabalho.

Aos meus velhos amigos Jonatas Bastos e Leandro Oliveira por me incentivarem a fazer um mestrado e por todo o apoio dado para que eu conseguisse uma vaga na UFPE.

Agradeço à Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) pelo suporte financeiro dados à minha pesquisa, o qual possibilitou a realização deste trabalho.

Finalmente, gostaria de agradecer a todos aqueles que colaboraram direta ou indiretamente na realização deste trabalho

...

Resumo

O “universo digital” cresce de forma exponencial e a previsão é que em 2015 chegue a quase 8 zettabytes. É cada vez mais complexo processar, armazenar, gerenciar, garantir a segurança e disponibilizar estas informações. O armazenamento em nuvem é apontado como melhor solução para lidar com estes problemas e muitos sistemas têm sido desenvolvidos com este fim. Grande parte destes sistemas tiram proveito da arquitetura *peer-to-peer* (p2p) para evitar gargalos, pontos únicos de falha e conseguir escalar de forma fácil. A maioria dos sistemas de armazenamento em nuvem existentes não permite que o usuário faça buscas mais complexas, por exemplo, levando em consideração o conteúdo dos arquivos ou informações presente nos metadados. Com isso essa pesquisa tem como objetivo propor uma abordagem para extração e indexação de conteúdo dos arquivos, de forma que o sistema permita que sejam realizadas buscas *full-text* baseadas no conteúdo dos arquivos em sistemas de armazenamento em nuvem com arquitetura *peer-to-peer*. Além disso, essa pesquisa também tem como objetivo investigar uma maneira eficiente para armazenar o índice de busca dentro da arquitetura P2P, de forma que seja possível replicar as informações com facilidade, permita elasticidade, garanta disponibilidade e, com isso, permitir que as buscas encontrem a maior quantidade de resultados relevantes possível.

Palavras-chave: Recuperação de Informação, Computação em Nuvem, Armazenamento em Nuvem

Abstract

The "digital universe"grows exponentially and is expected in 2015 will reach almost 8 zettabytes. It is becoming more complex process, store, manage, ensure safety and provide this information. The cloud storage is pointed as the best solution to deal with these problems and many systems have been developed for this purpose. Most of these systems take advantage of the peer-to-peer (p2p) architecture to avoid bottlenecks, single points of failure and and to scale easily. Most cloud storage systems existing does not allow the user to make more complex searches, eg, taking into account the contents of the files or information on metadata. Hence this research aims to propose an approach to extract and index the contents of files, such the system allows full-text searches based on the content of files in cloud storage systems based on peer-to-peer architecture. In addition, this research also aims to investigate an efficient way to store the search index within the P2P architecture, so that it is possible to replicate the information with ease, allowing elasticity, ensures availability and thereby allow the searches bring as many relevant results possible. In addition, this research also aims to investigate an efficient way to store search index within the P2P architecture, such it is possible to replicate the information with ease, allowing elasticity, ensures availability and, thus, allow searches to find as many relevant results possible.

Keywords: Information Retrieval, Cloud Computing, Cloud Storage

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

A cada dia se produz mais informação e, geralmente, estas informações são armazenadas em meios digitais. O tamanho desse "universo digital" cresce de forma exponencial. Segundo relatório publicado pela EMC Corporation[1] [Gantz and Reinsel, 2011], em 2005, o volume de arquivos chegou a 130 exabytes[2]; em 2010, superou 1 *zettabyte* e a previsão é que em 2015 chegue a quase 8 zettabytes[3], conforme mostrado na Figura 1.1.

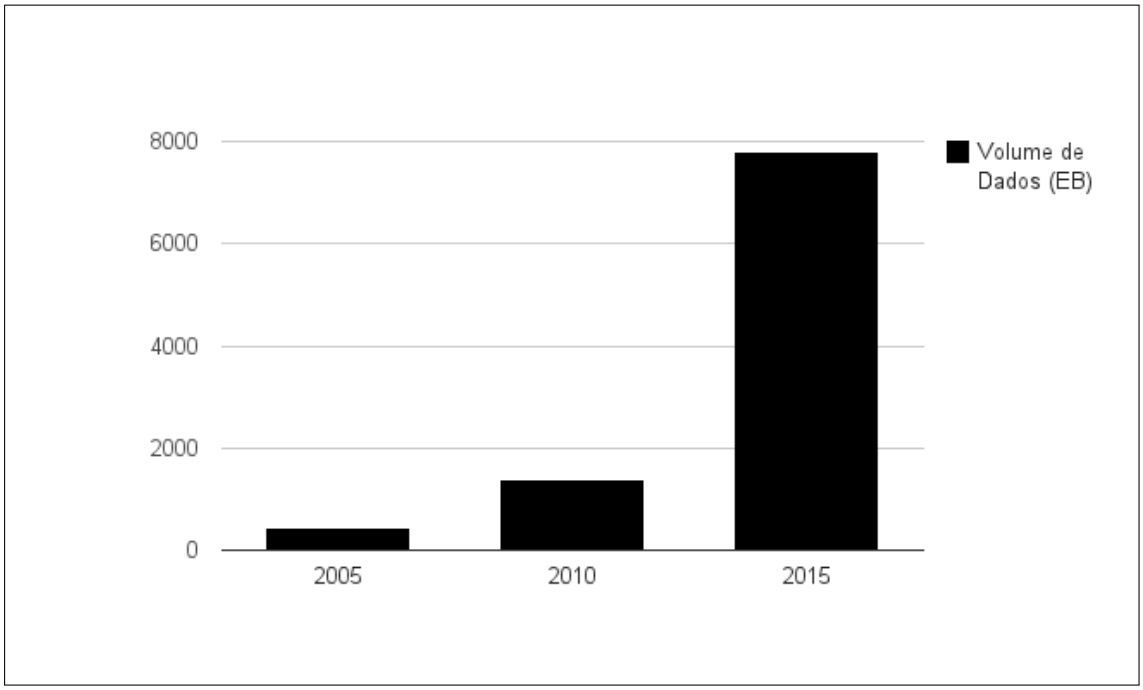

Figura 1.1 Gráfico mostrando a expectativa de crescimento da quantidade de arquivos gerados. Baseado em [Gantz and Reinsel, 2011].

Este "universo digital" citado em [Gantz and Reinsel, 2011] expande e se torna cada vez mais complexo processar, armazenar, gerenciar, garantir a segurança e disponibilizar estas informações. Somente a demanda por armazenamento cresceu mais de 50% nos

[1] http://www.emc.com Acessado em: 07/08/2013
[2] aproximadamente, 1.000.000.000 GB
[3] aproximadamente, 1.000.000.000.000 GB

últimos anos [Kaplan et al., 2008]. Para lidar com essas demandas, segundo [Gantz and Reinsel, 2011], serão necessárias habilidades, experiência e recursos, como capacidade de armazenamento, processamento, que rapidamente se tornarão escassos, além de uma nova infraestrutura de Tecnologia da Informação escalável e flexível. Uma opção que atende a estes requisitos é a computação em nuvens.

A utilização da computação em nuvens permite que os usuário possam alocar recursos dinamicamente, de acordo com sua necessidade. Entre esses recursos está o de armazenamento, o qual provê recursos e serviços de armazenamento baseados em servidores remotos que utilizam os princípios da computação em nuvem. A utilização da computação em nuvens para armazenamento permite que serviços de armazenamento sejam disponibilizados a um baixo custo [Zeng et al., 2009].

Diversos sistemas de armazenamento em nuvem dividem os arquivos em pedaços menores, chamados *chunks*, antes de armazená-los. Esta estratégia permite que seja efetuada uma replicação de dados mais eficiente e menos custosa, pois somente serão enviados pequenas partes do arquivo e não o arquivo completo. Além disso, como os arquivos são armazenados apenas de forma particionada, isto garante uma maior segurança dos dados [Rabin, 1989].

Quando é preciso lidar com um grande conjunto de arquivos, podemos utilizar técnicas de recuperação de informação para facilitar o processo de localização. Para isto, é necessário que os arquivos sejam indexados e um índice de busca seja gerado. A partir daí, um sistema de recuperação de informação (RI) poderá localizar os arquivos que são mais relevantes para a necessidade do usuário [Manning et al., 2008].

A necessidade de informação do usuário é representada através de sua expressão de busca, que pode ser especificada em linguagem natural (mesmos termos usados pelo autor do documento) ou através de uma linguagem artificial (termos determinados pelos desenvolvedores do sistema de RI), e deve resultar na recuperação de um número de arquivos que possibilite a verificação de cada um deles a fim de selecionar os que são úteis. A principal dificuldade do usuário está em predizer, por meio de uma expressão de busca, as palavras ou expressões que foram usadas para representar os arquivos e que satisfarão sua necessidade. Com o aumento da capacidade de armazenamento dos computadores, muitos sistemas de RI conseguem indexar o texto integral dos arquivos. Esta abordagem é conhecida como indexação *full-text* e é muito utilizada em grandes engenhos de busca, como: Google[4], Yahoo[5], Microsoft Bing[6], entre outros, e tem como objetivo permitir que o usuário possa localizar informações existentes no conteúdo dos arquivos, além de permitir a utilização de linguagem natural na expressão de busca.

1.1 Motivação

Os sistemas de armazenamento em nuvem mais populares, como: Amazon S3, Dropbox, SugarSync, Wuala, entre outros, somente permitem que o usuário localize

[4]http://www.google.com Acessado em 21/08/2013

[5]http://www.yahoo.com Acessado em 21/08/2013

[6]http://www.bing.com Acessado em 21/08/2013

algum arquivo pelo título do arquivo armazenado. Conforme discutido em [Kostoff, 2010] e [Berrios, 2000] a utilização de consultas *full-text* baseadas no conteúdo do arquivo podem aumentar a quantidade de resultados relevantes retornados em relação à consultas baseadas no título e palavras-chave. Além disso, pelo fato de ser possível consultar o conteúdo integral do arquivo, as consultas podem ser realizadas utilizando linguagem natural. Por exemplo, caso o usuário esteja escrevendo um artigo sobre o banco de dados Cassandra[7] e não consiga se lembrar do nome que usou para salvar o arquivo, mas sabe que o texto escrito possui a frase "O Cassandra é um banco de dados NoSQL". Utilizando consultas *full-text* baseadas no conteúdo, será possível utilizar esta frase para localizar o arquivo salvo, e não somente o nome do arquivo.

Para que seja possível realizar consultas *full-text* baseada em conteúdo é necessário que o conteúdo do arquivo seja extraído e, juntamente com os metadados, indexado para formar o índice de busca. Esta operação de indexação pode ser muito custosa caso seja realizada após a conclusão do armazenamento, já que muitos sistemas de armazenamento em nuvem armazenam somente os *chunks* dos arquivos, sendo necessário, assim, uma abordagem que possibilite que o conteúdo dos arquivos seja extraído e indexado no momento em que o usuário efetua o *backup* do arquivo no sistema de armazenamento.

Tomando isto como base, este trabalho propõe uma abordagem para permitir indexação e consultas *full-text* baseadas no conteúdo dos arquivos indexados com o intuito de melhorar os resultados de busca de arquivos em um sistema de armazenamento em nuvem. A abordagem será desenvolvida de forma que seja possível ser implantada em um sistema de armazenamento, no qual os arquivos sejam armazenados em *chunks*. Esta abordagem será implantada em um sistema real de armazenamento de arquivos em nuvem, o qual funciona sobre uma arquitetura *peer-to-peer*.

1.2 Definição do Problema

A maioria dos sistemas de armazenamento em nuvem somente permitem que consultas sejam feitas pelo título do arquivo. Visto que a utilização de consultas *full-text* permite que o usuário utilize uma linguagem natural nos termos de busca e isto pode aumentar a qualidade dos resultados retornados, o problema que esta dissertação se propõe a resolver pode ser definido como:

Os resultados retornados em uma busca por arquivos em um sistema de armazenamento em nuvem podem ser melhorados utilizando indexação e busca *full-text* baseadas em conteúdo?

[7] http://cassandra.apache.org/ Acessado em 21/08/2013

1.3 Solução Proposta

Esta dissertação tem como **objetivo geral** propor uma abordagem para realizar indexação e consultas *full-text* baseadas no conteúdo de arquivos para sistemas de armazenamento em nuvem.

Para o desenvolvimento desta pesquisa, os seguintes itens são considerados como **objetivos específicos**:

- **Identificar as propostas adotadas para recuperação de informação hoje pelos sistemas de armazenamento em nuvem** para analisar quais informações são armazenadas em forma de metadados e quais as opções de consultas disponibilizadas.
- **Realizar um levantamento de requisitos para as funcionalidades da abordagem proposta** de forma a tornar a abordagem viável para ser utilizada em um sistema real de armazenamento em nuvem.
- **Definir a estrutura dos metadados** que serão utilizados na abordagem proposta.
- **Definir e implementar uma solução para indexar arquivos**, de forma que seja possível implementá-la em um sistema real de armazenamento em nuvem, onde este guarda somente os *chunks* dos mesmos.
- **Definir e implementar uma solução para realizadar consultas *full-text***, levando em consideração a arquitetura na qual o sistema de armazenamento em nuvem foi implementado.
- **Realizar um estudo experimental** para avaliar a abordagem desenvolvida e o impacto de sua implementação e utilização.

1.4 Fora do Escopo

Alguns assuntos foram excluídos do escopo de pesquisa desta dissertação. São eles:

- **Atributos relacionados com a segurança do índice gerado na indexação ou dos arquivos armazenados:** O índice gerado na indexação contém informações importantes dos arquivos indexados, como o conteúdo, áreas de interesse do usuário, categoria, entre outros. Entretanto, não faz parte do escopo deste trabalho analisar qual a forma mais segura de armazenar o índice gerado e nem os arquivos armazenados.
- **Técnicas para evitar *flooding* em uma rede *peer-to-peer***: A abordagem proposta utiliza, como uma das estratégias possíveis de busca, a consulta ao índice de outros *peers* através da rede *peer-to-peer* formada. Se a quantidade de mensagens trocadas na rede for muito grande, isto pode gerar um problema conhecido com *flooding*. Esta pesquisa não aborda as diversas técnicas disponíveis para evitar este tipo de problema.

- **Modelo probabilístico para recuperação de informação**: Este modelo possibilita recuperar informação através de cálculos probabilísticos, usando para tal a teoria da probabilidade. Este modelo não é analisado nesta dissertação.
- **Indexação do conteúdo de arquivos de áudio e video**: Há diversas propostas na literatura propondo abordagens para indexação do conteúdo de arquivos de áudio, vídeo e imagens. Entretanto estas abordagens não serão analisadas no escopo deste trabalho.

1.5 Estrutura da Dissertação

O restante desta dissertação está organizada da seguinte forma:

- Capítulo 2 apresenta uma contextualização e os trabalhos relacionados com recuperação de informação em sistemas de armazenamento em nuvem.
- Capítulo 3 descreve a abordagem proposta, incluindo os requisitos, arquitetura e detalhes das implementações realizadas.
- Capítulo 4 apresenta um experimento com a abordagem proposta implementada.
- Capítulo 5 apresenta a conclusão do trabalho e aponta alguns possíveis trabalhos futuros.

2 Contextualização e Trabalhos Relacionados

Neste capítulo serão apresentados os conceitos de recuperação de informação e computação em nuvens, conceitos-chave desta dissertação, bem como alguns trabalhos relacionados com recuperação de informação em sistemas de armazenamento em nuvem.

2.1 Recuperação de Informação

Um sistema de recuperação de informação (RI) tem o objetivo de localizar a informação que é relevante para a consulta do usuário. Um sistema de RI tipicamente realiza consultas em coleções de dados não estruturados ou semi-estruturados (por exemplo, páginas *web*, documentos, imagens, vídeos, etc.). A necessidade de um sistema de RI ocorre quando um conjunto de dados atinge um tamanho que as técnicas tradicionais de catalogação já não permitem que os arquivos sejam encontrados facilmente. Para que isto seja possível, é necessário que os arquivos sejam indexados e um índice de busca seja gerado [Manning et al., 2008].

Os sistemas de RI devem representar o conteúdo dos arquivos e apresentá-los ao usuário de uma maneira que lhe permita uma rápida seleção dos itens que satisfazem total ou parcialmente à sua necessidade de informação, formalizada através da uma expressão de busca. O processo de representação busca descrever ou identificar cada arquivo através de seu conteúdo. Tal representação geralmente é realizada através do processo de indexação. Durante a indexação são extraídos conceitos do documento através da análise de seu conteúdo e traduzidos em termos de uma linguagem de indexação, gerando um índice onde serão realizadas as consultas [Hiemstra, 2009; Lu, 2007].

Um sistema de RI geralmente é desenvolvido com o objetivo de melhorar a eficiência e eficácia da tarefa de recuperação. A eficiência normalmente é calculada em termos computacionais (desempenho, tempo de CPU, etc) e a eficácia é dada, principalmente, pela taxa de precisão e *recall* [Baeza-Yates and Ribeiro-Neto, 1999]. Essas medidas são usadas para avaliar o desempenho de um modelo de RI e para fazer a comparação de performance entre diferentes modelos, os quais serão descritos na próxima Seção.

2.1.1 Modelos de Recuperação de Informação

De acordo com [Baeza-Yates and Ribeiro-Neto, 1999] um modelo de RI é composto por:

- Um conjunto de visões lógicas, ou representações, de documentos em uma coleção;
- Um conjunto de visões lógicas, ou representações, da informação requerida por usuários;
- Um *framework* para modelar documentos, consultas e suas relações;
- Uma função de ordenação que associa um número com uma consulta e um documento.

Nas próximas seções serão descritos os principais modelos modelos utilizados para recuperação de informação.

2.1.1.1 Modelo Booleano

O modelo Booleano é um dos mais antigo e simples modelo para RI. Ele se baseia na teoria dos conjuntos e na álgebra booleana [Baeza-Yates and Ribeiro-Neto, 1999]. As consultas são descritas por meio de expressões booleanas usando conectores lógicos: AND, OR e NOT e cada arquivo é representado por um conjunto de termos indexados, os quais simplesmente são palavras ou frases. Os arquivos recuperados são aqueles que satisfazem completamente a consulta do usuários. Aqueles que satisfazem parcialmente a consulta não são retornados. Como o resultado da recuperação de um arquivo é um valor binário, esse modelo não oferece um mecanismo de ordenação. A principal vantagem do modelo booleano se refere ao formalismo e à simplicidade. A principal desvantagem é a comparação exata entre consulta e arquivo, que pode levar à recuperação de poucos ou muitos arquivos [Baeza-Yates and Ribeiro-Neto, 1999; Hiemstra, 2009].

2.1.1.2 Modelo Espaço Vetorial

O modelo Espaço Vetorial tem o objetivo de resolver o principal problema do modelo booleano, que é a a incapacidade de retornar resultados que satisfaçam parcialmente a consulta e falta de ordenação dos arquivos retornados. No modelo Espaço Vetorial cada arquivo é representado por um vetor ou por uma lista de termos ordenados, por exemplo, pela frequência do termo no arquivo, em vez de apenas um conjunto de termos, como utiliza o modelo booleano [Baeza-Yates and Ribeiro-Neto, 1999]. Com isso, é possível, através deste modelo, associar um peso a cada termo de indexação ou da da expressão de busca. Esses pesos são utilizados para calcular o grau de similaridade entre a expressão de busca formulada pelo usuário e cada um dos arquivos. Desta forma, o modelo Espaço Vetorial retorna os arquivos ordenados de acordo com esta similaridade, ao invés de analisar se um arquivo é relevante ou não. As principais vantagens deste modelo são a possibilidade de retornar arquivos que satisfazem parcialmente à consulta e a possibilidade de ordenar os resultados. A principal desvantagem é que os termos de

indexação são mutualmente independentes; então este modelo não captura a semântica da consulta ou do arquivo [Manning et al., 2008].

2.1.1.3 Modelo Probabilístico

O modelo probabilístico tenta recuperar a informação através de cálculos probabilísticos, usando para tal a teoria da probabilidade. A ideia básica é a de recuperar os arquivos de acordo com a probabilidade de o arquivo ser relevante [Baeza-Yates and Ribeiro-Neto, 1999]. Existem diversas propostas baseadas no modelo probabilística como: [Rijsbergen, 1979], [Robertson, 1977], [Sparck Jones et al., 2000] e [Robertson and Zaragoza, 2009]

O modelo probabilístico não faz parte da proposta apresentada nesta dissertação. Portanto, as propostas relacionadas acima não serão detalhadas.

2.1.2 Recuperação de Informação *versus* Recuperação de Dados

Existe uma grande diferença entre os termos recuperação de informação e recuperação de dados e, como discutido em [Durão et al., 2008] e [Rijsbergen, 1979], eles são muitas vezes confundidos. Portanto, nesta Seção serão discutidas as características destas áreas a fim de entender as diferenças e situar a abordagem proposta em uma dessas.

De acordo com [Baeza-Yates and Ribeiro-Neto, 1999] e [Rijsbergen, 1979], a recuperação de dados tem como objetivo claro recuperar todos os objetos que satisfaçam condições definidas, por exemplo, em uma expressão regular ou em uma expressão de álgebra relacional. Assim, para um sistema de recuperação de dados, um único objeto recuperado erroneamente entre mil objetos recuperados significa fracasso total. Para um sistema de recuperação de informação, no entanto, os objetos recuperados podem ser imprecisos e pequenos erros são aceitáveis. A principal razão para essa diferença é que a recuperação da informação geralmente lida com texto de linguagem natural que nem sempre é bem estruturado e pode ser semanticamente ambíguo. Por outro lado, um sistema de recuperação de dados (como um banco de dados relacional) lida com dados que tem uma estrutura e semântica bem definida.

A Tabela 2.1, adaptada de [Rijsbergen, 1979], sumariza esta e outras diferenças existentes entre recuperação de dados e recuperação de informação.

Tabela 2.1 Tabela com diferenças entre recuperação de dados e recuperação de informação. Adaptada de [Rijsbergen, 1979]

	Recuperação de Dados	**Recuperação de Informação**
Correspondência	Exata	Parcialmente, melhor possível
Inferência	Dedução	Indução
Modelo	Determinístico	Probabilístico
Linguagem da Consulta	Artificial	Natural
Especificação da Consulta	Completa	Incompleta
Itens Desejados	Correspondentes	Relevantes
Sensibilidade a Erros	Sensível	Não-Sensível

Portanto, a solução proposta nesta dissertação integrará um sistema de recuperação de informação, pois esta não requer nenhum controle de sintaxe nas consultas, os resultados encontrados são baseados em relevância e fazem uso de dois modelos de recuperação de informação: modelo Espaço Vetorial e modelo Booleano.

2.2 Computação em Nuvem

O *National Institute of Standards and Technology* (NIST) define computação em nuvem como um modelo que permite que um conjunto de recursos computacionais possam ser fornecidos sob demanda de forma a permitir que os mesmos sejam fornecidos e liberados rapidamente com o mínimo de esforço de gestão ou interação do fornecedor [Mell and Grance, 2009]. Vaquero et al. [2009] define computação em nuvem como um grande conjunto de recursos virtualizados (como hardware, plataformas de desenvolvimento e/ou serviços) facilmente usáveis e acessíveis. A arquitetura da computação nas nuvens é comumente representada em camadas: Software como serviço, Plataforma como Serviço e Infraestrutura como serviço. Nesse modelo cada camada está construída sobre os serviços oferecidos pela camada de baixo [Lenk et al., 2009], conforme apresentado na Figura 2.1.

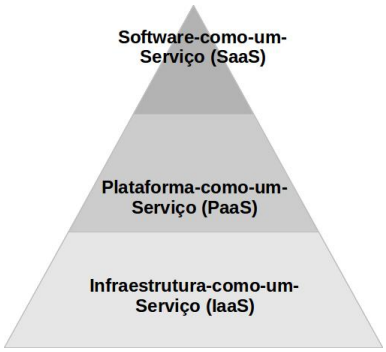

Figura 2.1 Arquitetura em camadas para computaçao nas nuvens

Infraestrutura como um serviço (*Infrastructure-as-a-Service* - IaaS) é o fornecimento de enormes recursos computacionais como: capacidade de processamento, armazenamento e conectividade. Tomando armazenamento como um exemplo, quando o usuário usa um serviço de armazenamento em computação em nuvem, ele somente paga a parte que foi efetivamente consumida sem comprar nenhum disco rígido ou até mesmo sem nem conhecer a localização dos seus dados armazenados. O exemplo mais conhecido de *IaaS* é o *Amazon Web Services* (*Amazon AWS*)[1], principalemente, através do *Amazon Elastic Compute Cloud* (*Amazon EC2*)[2].

Do ponto de vista do hardware, computação em nuvem trás três novos aspectos [Vogels, 2008]:

[1]http://aws.amazon.com/ Acessado em: 28/07/2013
[2]http://aws.amazon.com/ec2/ Acessado em: 28/07/2013

1. A ilusão de recursos computacionais infinitos disponíveis sob demanda, eliminando assim a necessidade dos usuários planejarem muito à frente para provisionamento de recursos.

2. A eliminação de um compromisso antecipado por parte dos usuários da nuvem, permitindo que as empresas comecem pequeno e aumente os recursos de hardware apenas quando há um aumento de suas necessidades.

3. A capacidade de pagar somente pelo que foi usado dos recursos computacionais (por exemplo, os processadores por hora e armazenamento por dia), e liberar recursos contratados facilmente quando não são mais necessários.

Plataforma como um serviço (*Platform-as-a-Service* - PaaS) geralmente abstrai a infraestrutura e dá suporte a um conjunto de interfaces de programação para aplicações em nuvem. É o meio entre o hardware e as aplicações. Devido a importância da plataforma, muitas grandes empresas querem aproveitar a oportunidade para tentar dominar essa área como a Microsoft fez na época do computador pessoal. Exemplo bem conhecidos são o *Google App Engine*[3], a plataforma de serviços do Microsoft Azure[4], o RackSpace[5].

Software como um serviço (*Software-as-a-Service* - SaaS) tem como objetivo substituir as aplicações que rodam nos computadores. Não há a necessidade de instalar ou rodar softwares especiais no computador para rodar um *SaaS*. Ao invés de comprar o software por um preço relativamente alto, o usuário paga pelo que é usado do *SaaS*, o que pode reduzir o preço total. O conceito de *SaaS* é atrativo e alguns softwares executam bem na nuvem, mas o atraso na rede é fatal para algumas aplicações, como sistemas de tempo real. Exemplos muito conhecidos de *SaaS* são o *SalesForce* [6], o Google Docs [7] e o Dropbox [8].

Como discutido em Armbrust et al. [2009], computação em nuvem refere-se tanto aos aplicativos entregues como serviços através da internet como ao hardware e software nos *datacenters* que oferecem esses serviços. Quando a nuvem é disponibilizada no formato *pay-as-you-go*, ou seja, pague pelo que você usa, para o público em geral, chamamos isso de uma nuvem pública. O termo nuvem privada é usado para se referir a *datacenters* internos de uma empresa ou outra organização, não disponibilizadas ao público em geral.

2.2.1 Armazenamento de Dados em Nuvem

Na Seção anterior mostramos que a utilização da computação em nuvens permite que os usuário possam alocar recursos dinamicamente, de acordo com sua necessidade. Entre esses recursos está o de armazenamento, o qual provê recursos e serviços de armazenamento baseados em servidores remotos que utilizam os princípios da computação em

[3]https://developers.google.com/appengine/ Acessado em: 28/07/2013
[4]http://www.windowsazure.com/ Acessado em: 28/07/2013
[5]http://www.rackspace.com/ Acessado em: 28/07/2013
[6]http://www.salesforce.com/ Acessado em: 28/07/2013
[7]http://docs.google.com Acessado em: 28/07/2013
[8]http://www.dropbox.com Acessado em: 28/07/2013

nuvem [Zeng et al., 2009]. Armazenamento em nuvem tem duas características básicas: a primeira trata da infra-estrutura da nuvem, a qual baseia-se em *clusters* de servidores baratos; a segunda tem o objetivo de, através dos *clusters* de servidores, armazenamento distribuído e redundância de dados, fazer múltiplas cópias dos dados armazenados para alcançar dois requisitos: alta escalabilidade e alta usabilidade. A alta escalabilidade significa que o armazenamento em nuvem pode ser dimensionado para um grande aglomerado com centenas de nós ou *peers* de processamento. Alta usabilidade significa que o armazenamento em nuvem pode tolerar falhas de nós e que estas falhas não afetam todos o sistema [Deng et al., 2010].

Com a popularização da computação em nuvem, cresceu o número de sistemas utilizados para armazenamento de dados em nuvem. O serviço oferecido por estes sistemas baseia-se no *software* que é executado em um *cluster* de servidores, os quais armazenam em seus discos rígidos os arquivos dos clientes, como mostrado na Figura 2.2. Normalmente, cada cliente possui um processo que controla a transferência de arquivos entre a máquina do usuário e os servidores na nuvem. Este processo também tem a função de se certificar que os dados enviados sejam espalhados por outros servidores no *cluster* e manter a sincronia dos dados na máquina do cliente e os dados armazenados, ou seja, novos dados gerados na máquina do cliente deverão ser salvos na nuvem e, caso os dados locais sejam perdidos, recuperá-los.

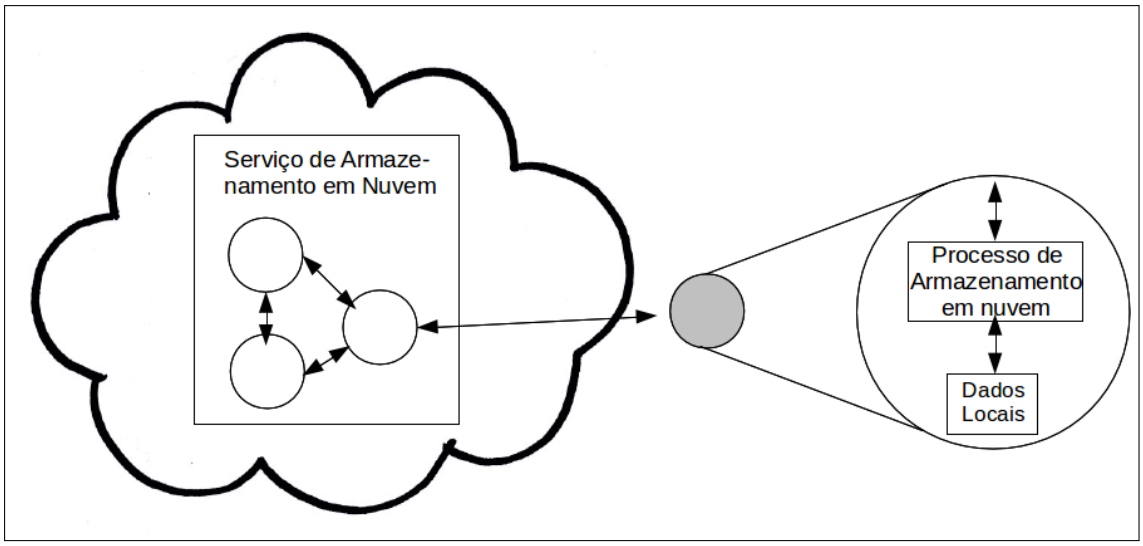

Figura 2.2 Arquitetura de um sistema de armazenamento em nuvem. Adaptado de [Peddemors and Spoor, 2010].

Diversos sistemas uniram os conceitos *peer-to-peer* com os conceitos de armazenamento em nuvem e passaram a oferecer soluções de armazenamento em nuvem baseadas em uma arquitetura *peer-to-peer*, como Wuala, BitTorrent Sync, Symform, Freenet, AeroFS.

Neste tipo de sistema, os nós que fazem parte da rede cooperam uns com os outros, através do compartilhamento de uma quota de espaço no disco rígido, e cada nó é, ao mesmo tempo, um consumir e produtor de dados. Como mostrado na Figura 2.3, cada nó

é composto por um processo que é responsável pelo envio dos dados locais para outros nós da rede e por receber os dados de outros *peers*.

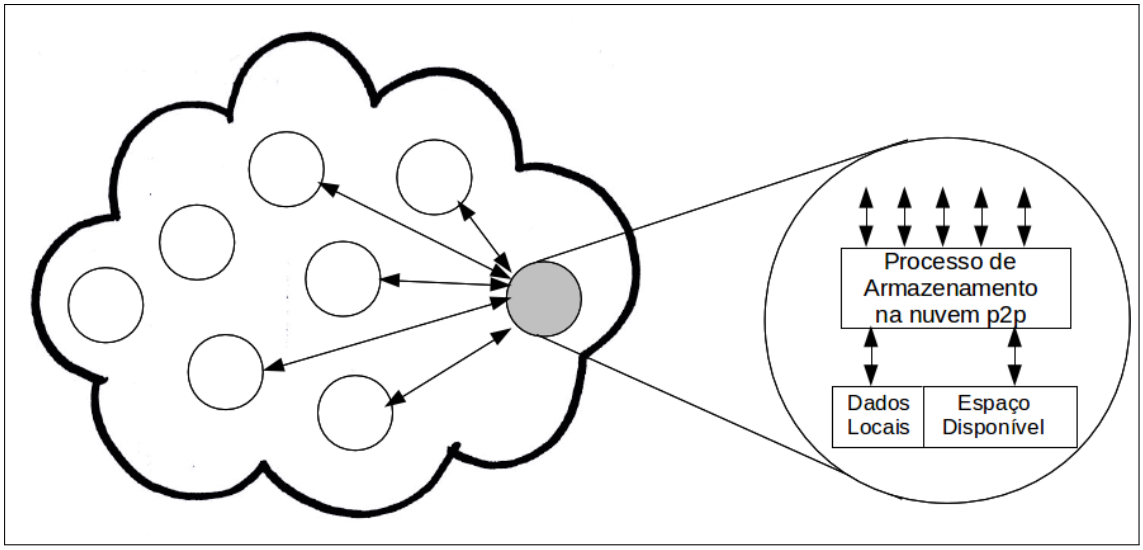

Figura 2.3 Arquitetura de um sistema de armazenamento em nuvem utilizando arquitetura P2P. Adaptado de [Peddemors and Spoor, 2010].

A arquitetura *peer-to-peer* permite que a rede formada tenha uma composição dinâmica, ou seja, *peer* podem entrar e sair da rede com facilidade e, para evitar que os dados armazenados em um *peer* sejam perdidos, é comum utilizar técnicas de armazenamento redundante [Datta et al., 2010]. Um efeito comum do armazenamento redundante de dados é a dispersão dos dados de forma geográfica (assumindo que a própria rede está distribuída geograficamente).

A redundância pode ser alcançada simplesmente através da replicação de arquivos para um número de *peers*, mas esta técnica é muito custosa, devido à necessidade da cópia integral do arquivo em cada *peers* escolhido e ao aumento do tráfego de rede para efetuar as cópias. Outra técnica muito utilizada é a *erasure code*, a qual é utilizada como técnica de correção de erros para compensar os erros ocorridos durante a transmissão de dados. Como representado na Figura 2.4, os dados originais são "quebrados" em pedaços menores e são criados fragmentos redundantes dos mesmos. Estes fragmentos são enviados para serem armazenados nos outros *peers*. Caso algum dos fragmentos armazenados seja perdido, o dados original ainda poderá ser re-montado à partir dos fragmentos restantes [Peterson and Weldon, 1972].

2.3 Trabalhos Relacionados

Nas seções anteriores apresentamos os principais conceitos relacionados com as áreas de recuperação de informação e computação em nuvem. Nesta Seção iremos analisar os trabalhos relacionados com recuperação de informação em sistemas de armazenamento

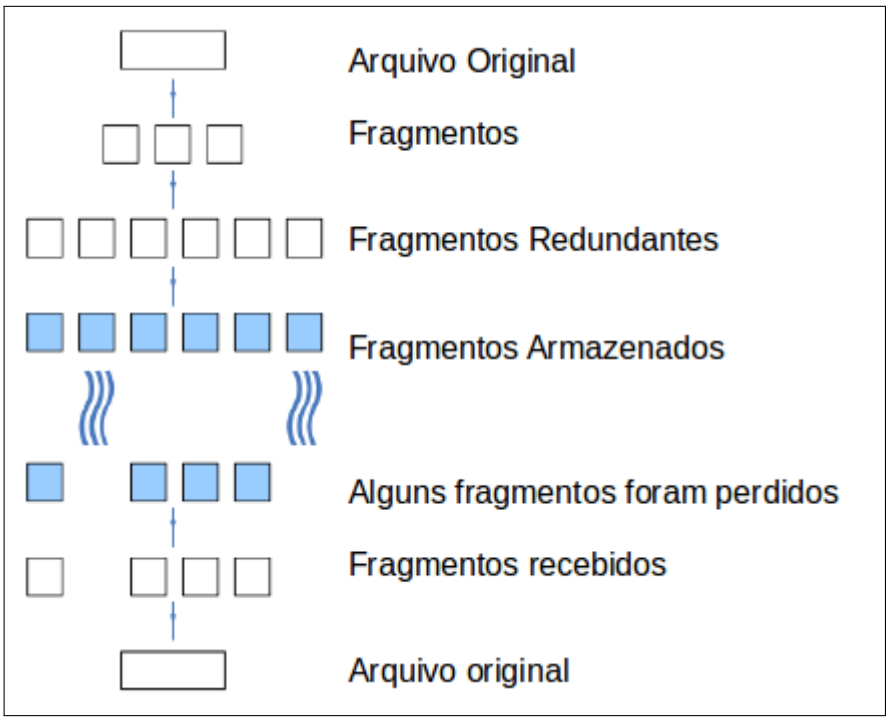

Figura 2.4 Representação do processo de para armazenamento e recuperação de dados utilizando *erasure code*. Adaptado de [Peddemors and Spoor, 2010].

em nuvem e mostrar como os principais sistema de armazenamento fazem para prover ao usuário uma maneira de efetuar consultas relacionadas com os arquivos armazenados.

2.3.1 Recuperação de Arquivos em Sistemas de Armazenamento em Nuvem

O Amazon S3[9] (*Simple Storage Systems*) é um sistema de armazenamento de objetos em nuvem oferecido pela Amazon Web Service[10] que funciona através de interfaces *web services* baseadas em *REST*, *SOAP* e *BitTorrent*.

O Amazon S3 foi projetado para fornecer escalabilidade, alta disponibilidade e baixa latência. Ele é formado por *buckets*, que são similares a diretórios ou *containers*. Cada objeto armazenado é composto pelo nome, um *blob* com o conteúdo (até 5 Gb) e os metadados do mesmo, composto por atributos que representam o tipo de dado armazenado, associação com usuários e permissões de acesso, datas de criação, acesso e modificação, localização de objetos relacionados, classificação e comentários feitos pelo usuário, rótulos de área, assunto e geolocalização[11].

O Amazon S3 só permite buscas limitadas a consultas simples baseadas no nome dos objetos e dentro de um único *bucket*, ou seja, ele não dá suporte para buscas baseadas nos

[9]http://aws.amazon.com/s3/, último acesso em 01/08/2013
[10]http://aws.amazon.com/, último acesso em 01/08/2013
[11]http://bit.ly/JeaAZg, último acesso em 01/08/2013

metadados e nem no conteúdo dos objetos armazenados [Palankar et al., 2008]. O Amazon S3 pode ser unido com o Amazon CloudSearch[12], que é um serviço da Amazon para inicializar, gerenciar e escalar uma solução de busca. O Amazon CloudSearch permite que arquivos já armazenados no S3 ou em algum banco de dados como DinamoDB[13] sejam indexados.

Entretanto, para utilizar o Amazon CloudSearch é necessário que o sistema esteja sendo executado na infra-estrutura da Amazon, ou seja, não é possível acoplar este serviço a um sistema externo. Além disso, para utilizar o Amazon CloudSearch seria necessário ter o arquivo completo armazenado para que o mesmo possa extrair os metadados necessários. Desta forma, não seria possível utilizá-lo em um sistema que armazene os arquivo particionados em *chunks* utilizando, por exemplo, o *erasure code*.

XtreemFS[14] é um sistema de arquivos globalmente distribuído e replicado. Ele é desenvolvido como parte do projeto XtreemOS EU, o qual tem o objetivo de criar um sistema operacional *open source* tolerante a falhas para qualquer necessidade de armazenamento em ambientes de *grid* ou computação em nuvem. XtreemFS é baseado em objetos, com metadados e dados armazenados em diferentes tipos de nós. Além disso, ele é compatível com o padrão POSIX[15], e possui sistema de tratamento de falhas. Ele replica os objetos para garantir tolerância a falhas e faz *cache* dos dados e metadados para melhorar o desempenho no caso de alta latência [Hupfeld et al., 2008, 2007]. O XtreemFS faz a replicação a partir do arquivo completo, ao invés de utilizar as partes já divididas do arquivo. Isso faz com que haja um aumento considerável na comunicação interna para manter as réplicas sincronizadas. O XtreemFS utiliza o BabuDB [Stender et al., 2010] como banco de dados para armazenar os metadados do sistema de arquivos. Os atributos utilizados nos metadados incluem uma identificação única para arquivos e diretórios, nome, tipo, além de atributos relacionados a data de criação e modificação, tamanho do arquivo, localização do conteúdos do arquivos, localização das réplicas[16], dono, controle de acesso e atributos estendidos.

O XtreemFS não armazena nenhum atributo relacionado com o conteúdo dos arquivos armazenados. Desta forma, o mesmo só permite que consultas sejam realizadas baseadas nos nomes do arquivos, diretórios, tipo de arquivo, entre outros atributos dos metadados.

O BitTorrentSync[17] permite que seja realizado o *backup* de arquivos e diretórios. Este utiliza o BitTorrent[18], que já é utilizado há muito tempo para compartilhamento de arquivos através de *torrent*. O BitTorrent é um protocolo de transferência de arquivos que, juntamente, com alguns *websites* como Piratebay.org e Mininova.org e servidores *trackers* provê, atualmente, um dos maiores sistemas de compartilhamento de arquivos. No BitTorrent os arquivos são dividios em pedaços menores (*chunks*) e o *peer* passa a disponibilizar cada arquivo para que outros usuários possam realizar o *download* dos

[12] http://aws.amazon.com/cloudsearch/ Acessado em: 02/08/2013
[13] http://aws.amazon.com/dynamodb/ Acessado em 02/08/2013
[14] http://www.xtreemfs.org
[15] http://standards.ieee.org/develop/wg/POSIX.html Acessado em: 27/10/2013
[16] http://www.xtreemfs.org/arch.php, último acesso em 20/07/2013
[17] http://labs.bittorrent.com/experiments/sync.html Acessado em 22/08/2013
[18] http://www.bittorrent.com/ Acessado em 06/08/2013

mesmos. Quando outros *peers* obtêm os arquivos, esses passar a disponibilizá-los também. Os servidores *tracker* tem a função de manter uma lista atualizada de quais *peers* possuem cada *chunk*.

Para encontrar algum arquivo, o usuário precisa utilizar algum dos *websites*, os quais disponibilizam informações como: nome do arquivo, tamanho, número atual de *peers* realizando *download* e enviando *chunks*, nome do usuário que disponibilizou o arquivo e palavras-chave adicionar pelo usuário [Pouwelse et al., 2005].

O Box[19] é um sistema online de armazenamento em nuvem e permite, na versão paga, que buscas *full-text* baseadas em conteúdo possam ser realizadas nos arquivos com extensão Microsoft Word, Excel, PowerPoint, PDFs, TXT e CSV. Entretanto, não há disponível nenhuma informação sobre como os arquivos são armazenados, nem como a indexação e busca são realizadas

Outros sistemas de armazenamento em nuvem como Dropbox[20], SugarSync[21], Google Drive[22], Wuala[23] não disponibilizam informações relacionadas com quais atributos são utilizados nos metadados, o que dificulta uma análise técnica aprofundada sobre os mesmos, mas é possível dizer que os mesmos não possuem uma indexação que permita que consultas sejam realizadas baseadas no conteúdo dos arquivos armazenados e que nestes sistemas só é possível realizar consultas baseadas no título dos mesmos.

2.4 Sumário do Capítulo

Neste capítulo foi apresentada uma revisão dos conceitos de recuperação de informação e computação em nuvem. Além disso, foi detalhado o funcionamento de sistemas de armazenamento em nuvem, os conceitos envolvidos no armazenamento e alguns trabalhos relacionados baseados em sistemas de armazenamento em nuvem disponíveis no mercado.

A maioria os sistemas de armazenamento hoje desenvolvidos oferecem opções simples para indexação e realização de consultas, normalmente somente baseadas no título dos arquivos armazenados. Alguns outros sistemas (como o Amazon S3) permite que as buscas sejam realizadas baseadas nos metadados armazenados, entretanto estes não armazenam nenhum atributo relacionado com o conteúdo armazenado. A Amazon propôs recentemente o Amazon Cloud Search (ainda em beta), que permite que buscas mais avançadas sejam realizadas no Amazon S3, entre elas a busca *full-text*. Entretanto, como discutido anteriormente, esta solução é exclusiva para ser utilizada com os serviços da Amazon (Amazon S3, DinamoDB, etc). Portanto, ela não é adequada para ser utilizada em sistemas de armazenamento em nuvem utilizando, por exemplo, o *erasure code*, para particionar e replicar os arquivos antes de armazenar, e o armazenamento distribuído, possibilitado pela utilização de uma arquitetura P2P. Ainda há o problema da falta de

[19] http://box.net/ Acessado em 22/08/2013
[20] http://www.dropbox.com, Acessado em 01/08/2013
[21] http://www.sugarsync.com/, Acessado em 01/08/2013
[22] https://drive.google.com, Acessado em 01/08/2013
[23] http://www.wuala.com/, Acessado em 01/08/2013

detalhamento de algumas propostas. Por exemplo, o Box apresenta uma solução para buscas *full-text* baseadas em conteúdo, entretanto, por ser um sistema proprietário, não há informações detalhadas do seu funcionamento.

Desta forma, a quantidade de trabalhos relacionados com recuperação de informação em sistemas de armazenamento em nuvem é pequena. Mas é possível dizer que falta uma proposta para indexação e consultas *full-text* baseadas no conteúdo dos arquivos armazenados para este tipo de sistema e isto nos motivou a desenvolver este trabalho.

No próximo capítulo será apresentada a abordagem proposta por esta dissertação, bem como os requisitos atendidos e a arquitetura da mesma.

3

Uma Abordagem para Indexação e Buscas *Full-Text* Baseadas em Conteúdo em Sistemas de Armazenamento em Nuvem

Neste capítulo será apresentada a abordagem desenvolvida nesta dissertação. Além disso, apresentamos também o Ustore, um sistema de armazenamento em nuvem privadas, no qual a abordagem foi implementada. Serão apresentados também os requisitos elicitados que esta abordagem se propõe a resolver, bem como detalhes de como foram implementados.

3.1 Ustore

O Ustore é uma ferramenta de armazenamento em nuvem baseada em uma arquitetura P2P híbrida que tem como objetivo armazenar dados com baixo custo e de forma que os mesmos não se tornem indisponíveis com eventuais problemas na rede [Assad et al., 2012]. Os dados a serem armazenados são quebrados em pedaços menores de tamanho pré-definido, chamados de *chunks*. Os *chunks* são armazenados em outros *peers* da rede, utilizando para isto os recursos ociosos disponíveis na rede. Estes *peers* podem, por exemplo, ser computadores da própria empresa utilizados para outros fins, mas que possuam espaço livre em disco suficiente para serem compartilhados. A utilização deste modelo faz com que o Ustore garanta um baixo custo para o armazenamento.

3.1.1 Arquitetura do Ustore

A arquitetura do Ustore consiste de uma arquitetura P2P híbrida em três camadas, onde há *peers* representando papéis distintos compondo a solução final. Os *peers* são agrupados em federações de dados, o que traz diversas vantagens, como: minimizar a sobrecarga na rede, em cada *peer* e reduzir a quantidade de mensagens trocadas. Este agrupamento permite uma maior escalabilidade do sistema, já que não há limites para a quantidade de federações criadas [Durão et al., 2013; Silva et al., 2012a].

A comunicação entre as entidade internas do sistema é feita através da plataforma JXTA. O JXTA[1] é um projeto de software livre de protocolos P2P baseados em mensagens XML para o desenvolvimento de aplicativos distribuídos, permitindo que qualquer dispositivo conectado em uma rede, independente de sua plataforma, natureza, ou protocolo de rede possa interagir, compartilhar recursos, e formar uma rede distribuída, descentralizada e cooperativa [Heiss, 2005]. O JXTA tem como objetivo encapsular funcionalidades e serviços comuns, escondendo a complexidade das implementações para o desenvolvedor. Na Figura 3.1 está representada a arquitetura do JXTA, na qual os *peers*, embora hierarquicamente semelhantes, podem possuir capacidade computacional distintas, já que uma das grandes vantagens de JXTA é garantir o uso racional e adequado dos recursos sem abrir mão da portabilidade.

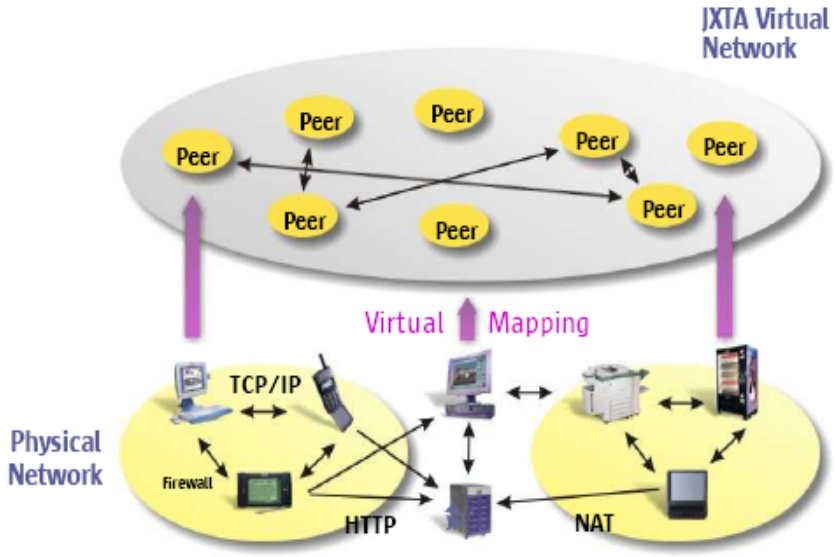

Figura 3.1 Representação da arquitetura do JXTA [Heiss, 2005]

Cada *peer* JXTA cria uma rede sobreposta virtual, permitindo a interação com outros pares normalmente inacessíveis, como aqueles protegidos por dispositivos reguladores de tráfego (como *firewalls*) ou que utilizem outro tipo de transporte de rede. Estes *peers* também podem ser organizados em grupos de uma forma descentralizada [Barolli and Xhafa, 2011]. Além disso, o JXTA desponibiliza recursos de rede, os quais possuem uma identificação única e constante, de forma a poderem mudar seu endereço de localização livremente sem perder a identificação e define um protocolo para as operações principais de uma rede P2P, como descobrimento de serviços, mensagens e organização de grupos. Dessa forma, o JXTA permite que cada *peer*, independente da plataforma de *software* e *hardware* utilizada, possa se conectar em uma rede P2P [Gong and Others, 2001].

A Figura 3.2 mostra os tipos de *peers* do Ustore [Silva et al., 2012b]:

[1]http://java.net/projects/jxta, último acesso em 30/05/2013

- **Cliente Ustore:** possui os dados a serem salvos e disponibiliza o espaço livre para armazenar *chunks*;
- **Servidor Ustore:** disponibiliza os serviços para serem utilizados pelos clientes e compõem as federações.
- **Super *peer*:** informa ao cliente, ao entrar na rede, onde estão os servidores. Também funciona como peer *relay*, ou seja, ele permite a troca de mensagens entre clientes que estejam em redes diferentes.

Cada tipo de *peer* do Ustore será descrito com mais detalhes nas próximas seções.

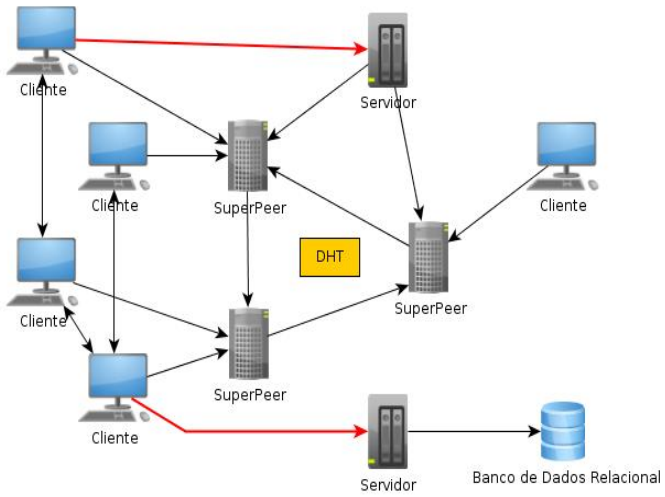

Figura 3.2 Representação da arquitetura do Ustore.

Cliente Ustore

Os clientes são aqueles que armazenam os *chunks* dos arquivos e, através deles, o usuário pode solicitar operações de *backup* e recuperação de arquivos. Cada cliente possui um horário de funcionamento estipulado inicialmente, que será utilizado para garantir a disponibilidade de dados no Ustore. Depois que um cliente se autentica na rede, ele pode salvar os dados que desejar. Estes dados são quebrados em *chunks* de tamanho pré-definido e enviados para que outros clientes que estão conectados os armazenem. Estes são escolhidos através de um algoritmo estatístico que localiza quais são os clientes que possuem o horário de funcionamento similar ao cliente inicial, desta forma o Ustore garante a disponibilidade dos dados no horário estipulado [Duarte, 2010]. Os *chunks* são espalhado na rede e somente o servidor possui as informações necessárias para remontá-los e, como forma de garantir uma maior disponibilidade, são replicados dentro da própria rede.

Servidor Ustore
Os *peers* servidores são aqueles que oferecem um conjunto (ou subconjunto) de serviços existentes. A definição de *peers* com funcionalidades específicas na rede opõem-se a algumas propostas para sistemas P2P, onde cada *peer* deveria ser capaz de desempenhar todos os papéis no sistema, promovendo a ideia de uma DHT (*Distributed Hash Table*) completa [Oliveira, 2007; Loest et al., 2009]. No entanto, a implementação utilizando uma DHT em sua essência é bastante custosa e de difícil escalabilidade. Sendo assim, a proposta adotada no Ustore é a criação de níveis hierárquicos que implementam serviços bem definidos e que podem crescer horizontalmente. Dentre os serviços disponibilizados pelos servidores, pode-se citar:

- **Autenticação**: usado para que cada cliente se autentique;
- **Disponibilidade**: permite checar a disponibilidade de cada peer;
- **Gerência de Diretórios**: utilizado para armazenamento e recuperação de diretórios inteiros;
- **Gerencia de Arquivos**: utilizado para armazenamento e recuperação de arquivos;
- **Busca por Dados**: procura por um conjunto de peers que obedeçam ao acordo de nível de serviço;

Super peer
Os super peers funcionam como elementos de referência para os demais componentes da arquitetura, sendo a porta de entrada para a participação de servidores, e clientes no sistema. O papel do super peer é definir as federações de dados quando cada cliente solicita conexão à rede. Para isto, os super peers devem ter sua localização previamente conhecida por todos os demais peers por meio de uma pré-configuração. Também é papel deste tipo de peer, escolher dinamicamente os clientes e servidores das federações baseando-se em um algoritmo de proximidade [Duarte, 2010]. O agrupamento em federações permite o crescimento elástico e garante a escalabilidade do sistema, pois não existe um limite para a quantidade de federações que podem ser criadas.

3.2 Requisitos

Os requisitos (funcionais e não-funcionais) propostos neste trabalho para o engenho de busca *full-text* são baseadas nos problemas enfrentados por outros engenhos de busca analisados no Capítulo 2. Além disso, os requisitos também são derivados da área de computação em nuvens e discussões no grupo ASSERT[2]. Neste contexto, um conjunto de requisitos funcionais e não-funcionais são apresentados nas seções seguintes.

[2]http://www.assertlab.com Acessado em: 26/06/2013

3.2.1 Requisitos Funcionais

- **Indexar no lado do cliente** - Muitos sistemas de armazenamento de arquivos em nuvem adotam a estratégia de não armazenar o arquivo inteiro e, sim, dividido em pedações menores, chamados de *chunks*. Devido a isto, o processo de indexação deve acontecer no cliente que está efetuando o *backup*, já que este será o único cliente com acesso ao arquivo inteiro.

- **Indexar vários formatos de arquivos** - A indexação deve ser feita em diversos tipos de arquivos, incluindo: arquivos de todo o pacote *Office*, arquivos de todo o pacote *OpenOffice*, arquivos com extensão pdf, arquivos de imagem, arquivos de vídeo.

- **Extrair e indexar conteúdo dos arquivos de texto** - Para os arquivos de texto (por exemplo, arquivos com extensão: pdf, doc, xls, txt), o sistema deve extrair e indexar também o conteúdo dos mesmos para que seja possível realizar a consultas *full-text* baseadas no conteúdo.

- **Realizar consultas *full-text*** - O sistema deve permitir que o usuário realize consultas do tipo *full-text* baseadas no conteúdo dos arquivos. O sistema deve retornar uma quantidade estipulada inicialmente de resultados, organizando-os pelo número de ocorrências dos termos escolhidos pelo usuário.

- **Realizar a indexação global-local** - O sistema deve realizar a indexação dos metadados em um índice global e em um local. O índice local permitirá que o cliente realize consultas somente entre seus próprios arquivos e também poderá responder a consultas de outros clientes. O índice global deverá ficar no Servidor de Busca e agregará o índice de todos os arquivos do sistema.

- **Realizar consultas locais e globais** - O sistema deve permitir que consultas *full-text* sejam realizadas no índice local do cliente e de outros clientes e no índice global.

- **Apresentar resultados da consulta para o usuário** - O resultado da consulta deve ser apresentado para o usuário classificados de acordo com o número de ocorrências dos termos utilizados. Além disso, o usuário deverá ter a opção de solicitar o *download* do arquivo a partir do resultado apresentado.

3.2.2 Requisitos Não-Funcionais

- **Extensibilidade** - Em geral, qualquer *software* deve levar em consideração o crescimento futuro. As restrições de arquitetura devem presumir a adição de novas funcionalidades e o nível de esforço necessário para implementá-las sem afetar às funcionalidades já sistema existentes. Por esta razão, a solução proposta deve ser bem estruturada, com módulos de baixo acoplamento para permitir uma fácil manutenção e extensão;

- **Elasticidade** - O sistema deve permitir que a solução consiga expandir e diminuir os recursos alocados de forma automática.
- **Replicação** - O sistema deve replicar os índices gerados como forma de garantir que não sejam perdidos em eventuais falhas.
- **Usabilidade** - O sistema deve fornecer uma interface gráfica com componentes intuitivos para realizar as funcionalidades.
- **Desempenho** - O desempenho normalmente envolve a análise do tempo de resposta. Em um sistema distribuído, além do poder de processamento, tráfico de rede, distribuição geográfica, entre outros atributos, exercem influência sobre o tempo de resposta.
- **Alta precisão e *recall*** - Este é um requisito essencial que deve ser considerado em qualquer mecanismo de busca. O desempenho de uma ferramenta de recuperação de informação é medido tendo em conta a precisão alcançada e as taxas de *recall*. Alta precisão é alcançada quando os elementos mais relevantes são retornados em uma pesquisa e alto *recall* é alcançado quando poucos elementos relevantes são deixados de fora do resultado.

A Tabela 3.1 sumariza a situação dos requisitos apresentados. Os requisitos na situação "Alcançado" significa que foram completamente desenvolvidos e avaliados. Aqueles na situação "Parcialmente Alcançado" foram desenvolvidos, entretanto não foram avaliados ou precisam de melhorias.

Tabela 3.1 Tabela com a situação dos requisitos propostos.

Requisito	**Situação**
Indexar no lado do cliente	Alcançado
Indexar vários formatos de arquivos	Alcançado
Extrair e indexar conteúdo dos arquivos de texto	Alcançado
Realizar consultas *full-text*	Alcançado
Realizar a indexação global-local	Alcançado
Realizar consultas locais e globais	Alcançado
Apresentar resultados da busca para o usuário	Alcançado
Extensibilidade	Parcialmente Alcançado
Elasticidade	Parcialmente Alcançado
Replicação	Parcialmente Alcançado
Usabilidade	Alcançado
Desempenho	Alcançado
Alta precisão e *recall*	Parcialmente Alcançado

3.3 Indexação e Consultas *Full-Text* Baseadas em Conteúdo

Na Seção anterior foram apresentados os requisitos que serão desenvolvidos por esta abordagem, cujo objetivo é permitir que consultas *full-text* baseadas em conteúdo sejam realizadas no Ustore. Nesta Seção será mostrado como a abordagem proposta extrai e indexa o conteúdo de cada arquivo e, após isso, serão mostradas as estratégias de busca adotadas no Ustore. Cada estratégia se refere à um tipo de busca diferente: local, através da rede e no Servidor de Busca.

3.3.1 Arquitetura

Arquitetura de *software* lida com a concepção e implementação da estrutura de alto nível do *software*. É o resultado da montagem de um certo número de elementos arquiteturais em algumas formas bem-definidas. A seguir é apresentada a visão de componentes do Ustore juntamente com os componentes da abordagem proposta.

3.3.1.1 Visão de Componentes

Os clientes do Ustore e o Servidor de Busca possuem componentes específicos para a realização de indexação e consultas *full-text*. Como mostrado na Figura 3.3, nos clientes, o componente responsável pela indexação se comunica diretamente com o componentes responsável pelo controle dos *backups* efetuados no sistema. Isto se justifica pelo fato de existir um vínculo entre estes procedimentos com operações que ocorrem paralelamente, como será mostrado mais adiante. Já o componente de busca (*Searcher*) se comunica diretamente com a interface gráfica do Ustore (*GUI*), onde o usuário pode informar os termos que deseja pesquisar e onde visualizará o resultado da consulta.

O Servidor de Busca possui somente os componentes de indexação e busca e disponibiliza duas interfaces para, respectivamente, receber informações para serem indexadas e receber os termos de busca.

3.3.2 Indexação *Full-Text*

Conforme mostrado na Figura 3.3 cada cliente e Servidor de Buscas do Ustore possui um componente responsável pela indexação. A Figura 3.4 mostra o diagrama de sequência do processo de indexação que se inicia quando o cliente informa ao servidor que um *backup* será iniciado.

A primeira mensagem enviada para o servidor faz com que algumas informações sobre o *backup* sejam registradas e, através do algoritmo de disponibilidade de peers[Duarte, 2010], descubra quais são os melhores clientes para os quais o cliente deverá enviar os *chunks* do arquivo. Após isto, o cliente divide o arquivo em *chunks* com tamanho pré-definido pelo usuário. Paralelamente ao envio dos *chunks* para os outros clientes, o componente de indexação é acionado e, primeiramente, identifica qual o tipo do arquivo que está sendo salvo. Isto permite que seja escolhido um *parser* adequado para possibilitar

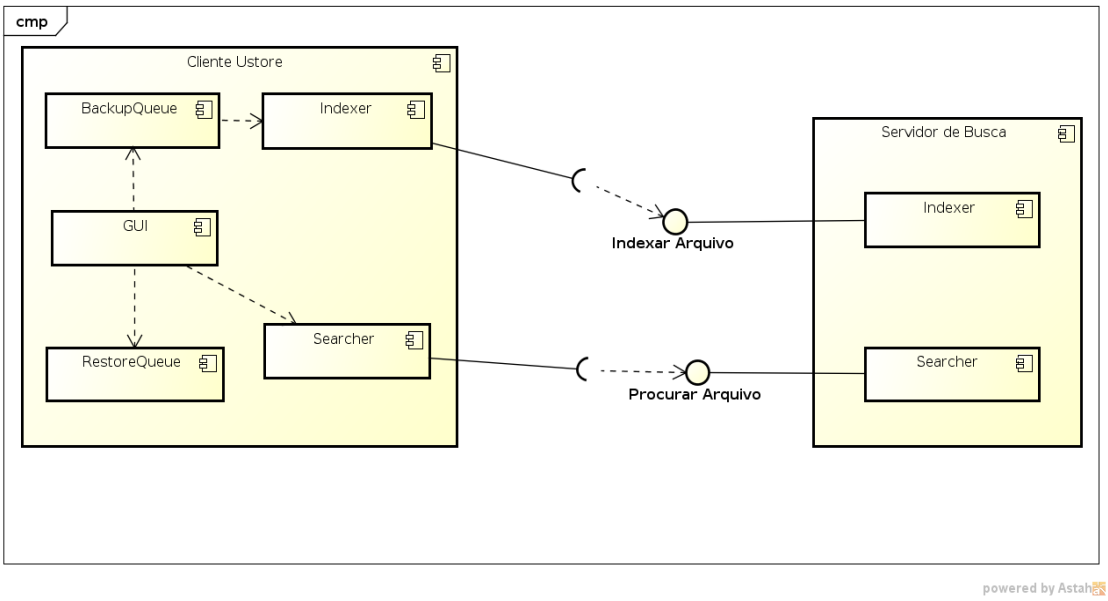

Figura 3.3 Diagrama de componentes do Ustore mostrando os componentes para indexação e busca.

a extração dos metadados juntamente com o conteúdo do mesmo. Estas informações são armazenadas no objeto *Index* para posterior indexação.

Metadados

Visando atender aos requisitos descritos, tornou-se necessário a implementação de um modelo de metadados. Para isto, o primeiro passo, conforme definido por [Steinacker et al., 2001], é a definição da sintaxe do mesmo, ou seja, quais atributos devem fazer parte da estrutura. Os atributos dos metadados foram definidos tendo como base os requisitos definidos anteriormente na Seção 3.2, e os trabalhos relacionados com computação em nuvens, recuperação de informação e metadados, discutidos no Capítulo 2.

Os atributos dos metadados extraídos durante a indexação estão descritos na Tabela 3.2.

Os metadados que serão indexados devem ser armazenados em dois locais, a saber:

- Localmente: Os metadados são indexados e armazenados na máquina onde a aplicação cliente está instalada;
- No Servidor de Buscas: O cliente que está realizando o *backup* envia os metadados para serem indexados e armazenados no Servidor de Buscas;

Conforme descrito na Seção 3.1, o Ustore divide os arquivos em *chunks* ainda no cliente que está efetuando o *backup* e estes *chunks* são enviados para outros clientes para serem armazenados. Desta forma, nem os clientes que recebem os *chunks* e nem o servidor do Ustore possuem o arquivo inteiro e, por isto, a extração de metadados e do conteúdo somente deverá ser executada pelo próprio cliente que efetua o *backup*, pois

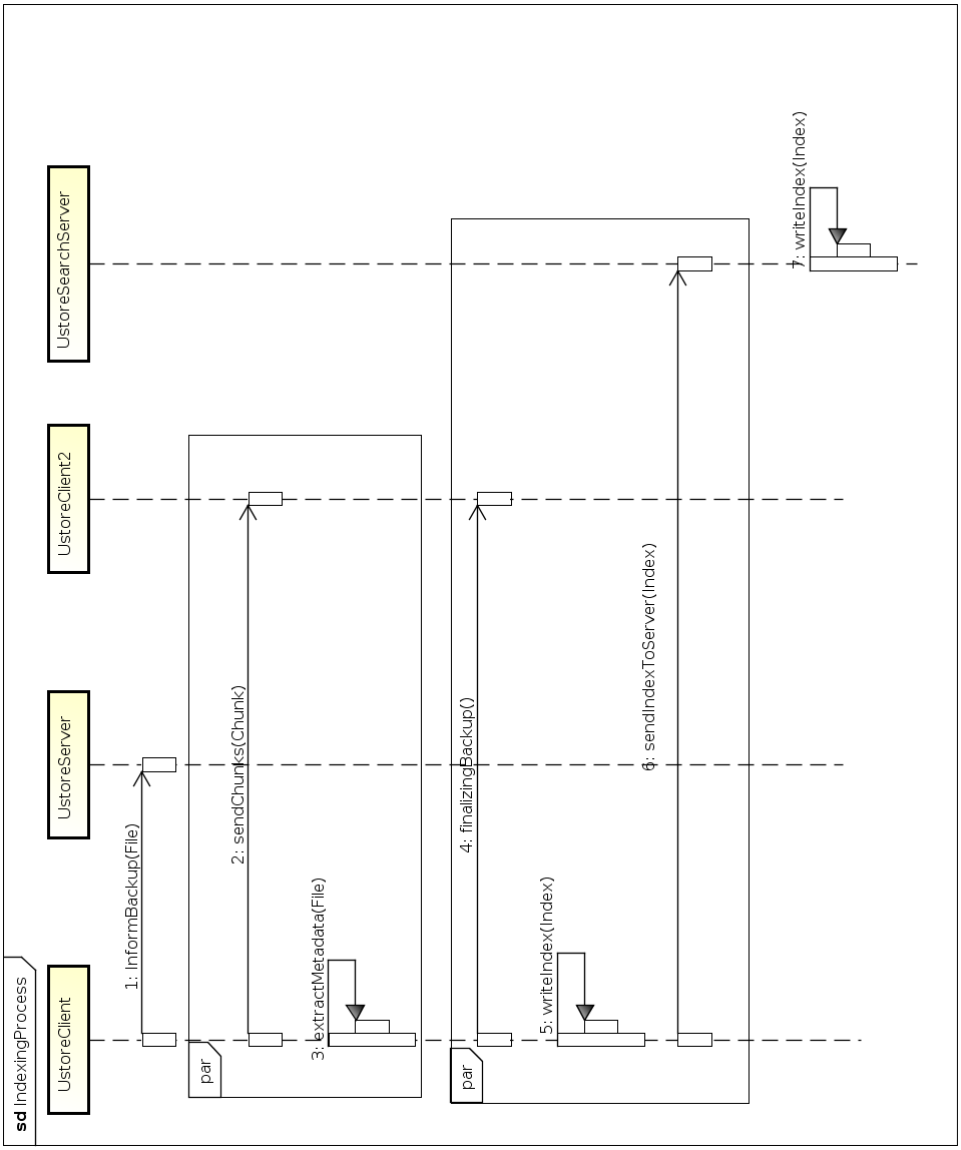

Figura 3.4 Diagrama de sequência mostrando o processo de indexação.

Tabela 3.2 Tabela de metadados indexados no Ustore.

Atributo	Descrição
FILE_NAME	Nome do arquivo
FILE_NAME_LOWER	Nome do arquivos (minúsculas)
SIZE	Tamanho do arquivo (Kbytes)
PATH	Caminho completo do arquivo
PATH_LOWER	Caminho completo do arquivo (minúsculas)
FILE_ID	ID do arquivo no banco de dados
CONTENT	Conteúdo do arquivo
LANGUANGE	Linguagem do arquivo
CATEGORY	Categoria do arquivo
USER_INTEREST	Áreas de interesse do usuário
AUTHENTICATION	Número de autenticação do usuário
DATE_CREATED	Data de criação do arquivo
DATE_MODIFIED	Data de modificação do arquivo
DATE_INDEXED	Data de indexação do arquivo

somente este possui o arquivo inteiro. Além disto, o componente de indexação também tem a responsabilidade de criar um índice local, como será descrito adiante.

A indexação foi realizada através do Apache Lucene[3]. A indexação processa os dados originais gerando uma estrutura de dados inter-relacionada eficiente para a pesquisa baseada em palavras-chave. O Lucene utiliza em seu índice a estrutura de dados chamada de índice invertido, onde cada termo adicionado possui uma referência para o arquivo que o contém. Pelo fato de o Lucene não possui um *parser* próprio, foi necessário utilizar uma outra biblioteca que tivesse essa capacidade. O *parser* é necessário para extrair as informações que compõem os metadados e, para isto, foi utilizado o Apache Tika[4], que é um detector e extrator de conteúdo de metadados e texto estruturado, podendo ser utilizado com arquivos de diversos formatos. Mais detalhes sobre o Apache Lucene e Apache Tika serão apresenados ao final do capítulo.

Indexação Local

A indexação dos metadados localmente acontece no próprio cliente que está efetuando o *backup* e acontece paralelamente à finalização do mesmo, conforme apresentado na Figura 3.4. Esta indexação tem a vantagem de funcionar independente de problemas na rede, entretanto, permite que o cliente consulte unicamente o índice existente no cliente. Estas consultas acontecem através de mensagens síncronas, pois o cliente tem acesso direto ao índice gerado. Outra desvantagem é que o índice existe unicamente neste cliente, o que significa que caso o cliente apresente algum problema que ocasione a perde deste índice, não será possível recuperá-lo, será necessário executar uma rotina para realizar o

[3]http://lucene.apache.org/ Acessado em 26/06/2013

[4]http://tika.apache.org/, último acesso em 21/05/2013

download de todos os arquivos e indexá-los novamente.

Indexação no Servidor de Buscas

O envio dos metadados para serem indexados no Servidor de Busca ocorre de forma paralela à finalização do *backup*, da mesma forma que a indexação local. Uma mensagem JXTA assíncrona é enviada através da rede para o Servidor de Busca. Quando esta chega ao servidor, os metadados que ela contém são indexados no índice global. A utilização de um índice global permite que o usuário faça uma consulta levando em consideração todos os arquivos públicos indexados pelo Ustore, mas o servidor pode se tornar um gargalo no sistema e se transformar em um ponto de falha. Por isto, foi adotada uma solução para garantir a escalabilidade dos servidores e a replicação de dados entre eles, que será descrita na próxima Seção.

Elasticidade e Replicação do Índice entre Servidores de Busca

Como o JXTA possui o recurso de auto-descobrimento de *peers* na rede, é possível criar um novo Servidor de Busca facilmente e integrá-lo à rede existente. Caso existam dois ou mais servidores de busca ativos, o cliente escolherá aleatoriamente um e enviará a mensagem para este. Após a indexação neste, o mesmo encaminhará a mensagem para outro servidor, também escolhido de forma aleatória, para que o conteúdo também seja indexado. Com este procedimento o Ustore garante a replicação do índice de busca global, descrito com mais detalhes na próxima Seção. O índice de busca gerado é a parte mais importante do sistema de busca. Ele é o único que possui os metadados dos arquivos indexados e por isto a replicação do mesmo é muito importante para garantir a disponibilidade de informações. No diagrama de atividade na Figura 3.5 está representado o processo de indexação e replicação do índice de busca implementado no Ustore, onde nota-se que para cada *backup* são realizadas três indexações em locais diferentes. A primeira ocorre no cliente que está efetuando o *backup*; a segunda, em um Servidor de Buscas escolhido aleatoriamente pelo cliente; e a terceira, em outro Servidor de Busca escolhido aleatoriamente pelo primeiro Servidor de Buscas escolhido; de forma que seja possível reduzir as chances de que o índice de busca seja perdido, juntamente com todos os metadados indexados, devido a alguma falha em uma das instâncias.

Existe ainda a possibilidade de diminuir a quantidade de servidores de busca existentes ou que algum servidor apresente falha. Nestes casos, o índice seria perdido e a informação não estaria mais adequadamente replicada. Para solucionar isto, é possível executar um algoritmo que faz a junção de dois ou mais índices de busca.

3.3.3 Estratégias para Consultas *Full-Text*

Na Figura 3.3 o componente responsável pela consulta interage diretamente com a GUI (*graphical user interface*). A Figura 3.6 mostra como se dá o processo de consulta envolvendo outros clientes do Ustore e o Servidor de Buscas.

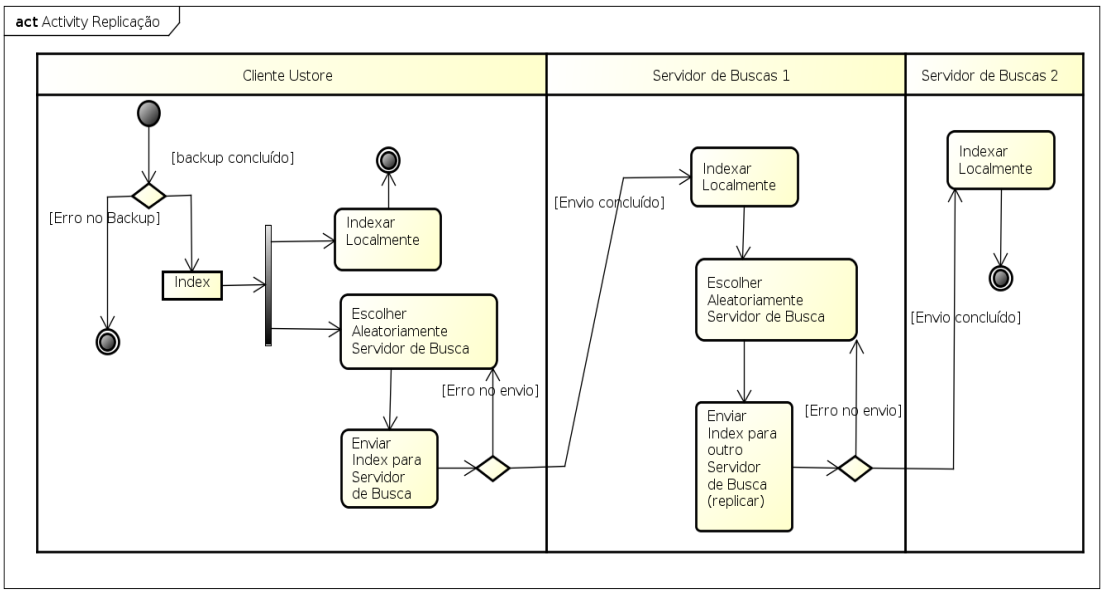

Figura 3.5 Diagrama de atividades representando o processo de indexação e replicação do índice de busca implementado no Ustore.

Na GUI o usuário informa quais os termos que deseja procurar nos arquivos armazenados no Ustore. Estes termos formam o objeto *SearchQuery*, juntamente com informações relativas ao cliente que o usuário está usando. No Ustore há três formas de se realizar uma consulta por informações, a saber:

- Busca local: esta consulta leva em consideração somente o índice armazenado no cliente em que o usuário está;
- Busca na rede: esta consulta leva em consideração os índices armazenados nos outros clientes e é realizada através de troca de mensagens JXTA pela rede P2P;
- Busca no Servidor de Busca: esta consulta leva em consideração o índice global armazenado no Servidor de Busca;

Em todos os casos de busca, o cliente ou servidor que recebe a requisição de busca deve realizar uma consulta em seu índice local e localizar entre os arquivos indexados quais possuem a maior quantidade de ocorrências do termo de busca passado. Após isto, os resultados encontrados são retornados para o cliente inicial e exibidos para o usuário, conforme apresentado na Figura 3.7.

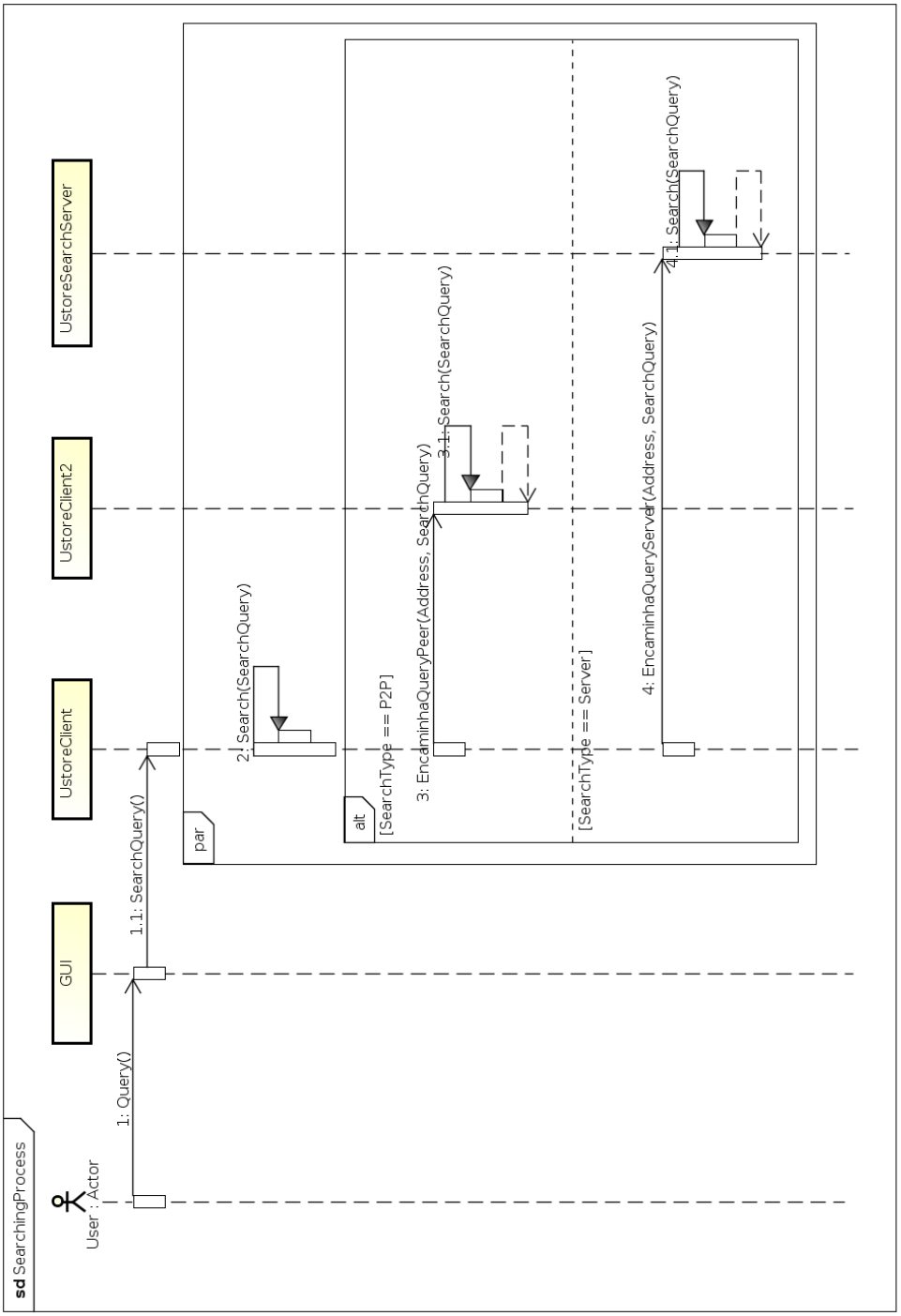

Figura 3.6 Diagrama de sequência mostrando o processo de busca.

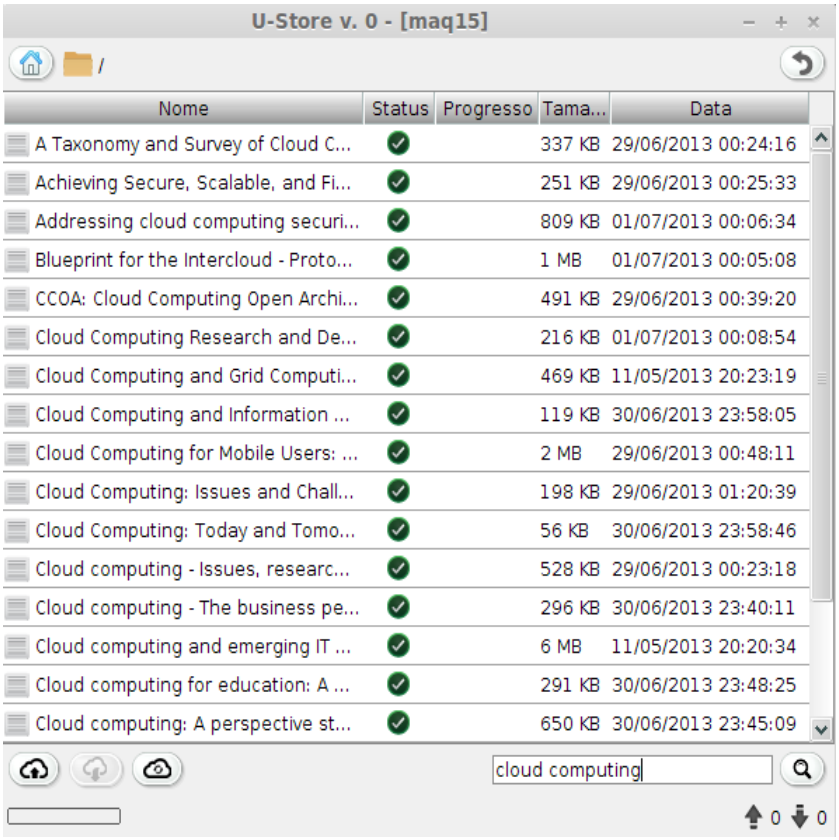

Figura 3.7 Tela do Ustore exibindo resultado da busca.

Caso o usuário opte por realizar a consulta utilizando os servidores e os mesmos estiverem inativos, o Ustore pode detectar este problema e passar a realizar a consulta através da rede. Caso nenhum outro cliente esteja conectado no momento da consulta (impossibilitando este tipo de busca), será possível fazer uma consulta utilizando unicamente o índice armazenado localmente. A existência de formas alternativas para realização de consultas, juntamente com a capacidade do Ustore de detectar estas falhas, faz com que o mesmo possua um mecanismo que garanta que uma busca sempre será realizada.

Consulta Local

A consulta local permite que o cliente do Ustore faça uma consulta de forma síncrona ao seu próprio índice. Esta consulta sempre irá funcionar, independente da existência de algum problema de conexão na rede. Porém, como não há consulta aos índices em outros clientes ou servidores, os resultados ficam restritos apenas aos arquivos indexados naquele cliente e que sejam públicos (em caso de arquivos pertencentes a outros usuários do sistema, mas indexados no mesmo cliente).

Consultas em outros clientes Ustore pela rede

O Ustore também permite que o usuário opte pela consulta através da rede; caso isto ocorra, o Ustore utilizará o recurso de auto-descobrimento do JXTA para descobrir quais clientes conectados na rede possuem o serviço de busca associado e encaminhará a consulta para eles. Para isto o Ustore utiliza mensagens JXTA assíncronas e, uma vez que a consulta chegue até um cliente, este realizará a consulta no seu índice local e retornará os resultados para o cliente que lhe encaminhou a consulta. Utilizando esta estratégia é possível analisar mais índices à procura de arquivos públicos que satisfaçam aos termos informados pelos usuários. Apesar disto, uma consulta implicará em enviar uma mensagem assíncrona para todos os clientes conectados naquele momento. Quanto maior a quantidade de clientes conectados, maior será a quantidade de mensagens enviadas, o que poderá ocasionar uma degradação do desempenho da rede, este problema é conhecido como *query flooding*. Há diversas propostas para evitar que este problema aconteça, entretanto não faz parte do escopo deste trabalho analisá-las.

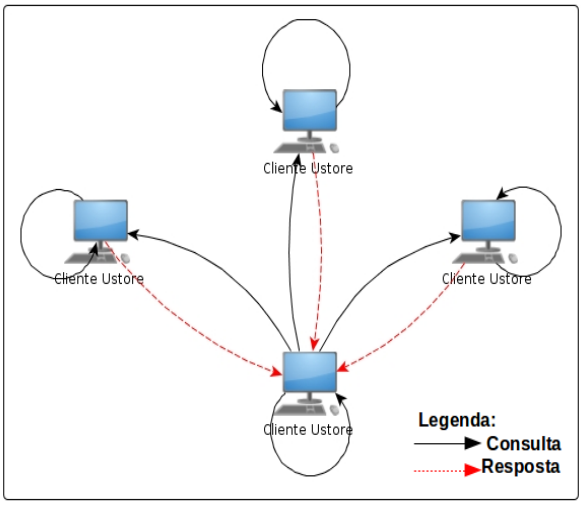

Figura 3.8 Representação da consulta através da rede P2P

Consultas no Servidor de Buscas

A última estratégia para consultas no Ustore é realizar a consulta em um ou mais servidores exclusivos para indexação e consultas. Nesta abordagem o Ustore utiliza o auto-descobrimento do JXTA para localizar um serviço específico que roda nos servidores e, caso exista mais de um, escolhe um aleatoriamente, e envia para o mesmo a consulta desejada. O servidor que recebe a consulta, faz uma consulta no seu índice local e, caso a quantidade de resultados não alcance um mínimo estipulado, este servidor encaminhará a consulta para um outro servidor que tenha o serviço disponível, recomeçando o ciclo. Este ciclo durará até que a quantidade mínima de resultados seja alcançada ou até que não exista mais servidores disponíveis não consultados. Após a finalização os resultados são retornados para o cliente inicial.

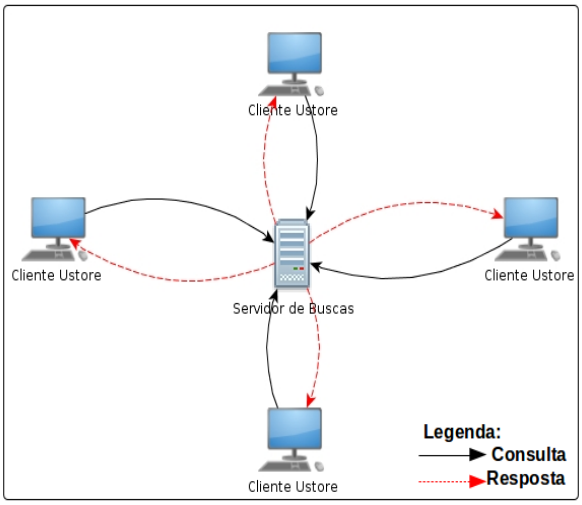

Figura 3.9 Representação da consulta utilizando o Servidor de Buscas

Com esta estratégia, as consultas são realizadas sobre um índice global, pois os servidores indexam os arquivos de todos os usuários. Como descrito anteriormente na Seção 3.3.2, os servidores são sempre escolhidos aleatoriamente, tanto para indexação quando para replicação do índice. Sendo assim, os servidores não possuem exatamente o mesmo conteúdo nos seus índices.

3.4 Detalhes da Implementação

Para implementação da abordagem proposta no Ustore foi utilizado o Apache Lucene versão 3.6 e Apache Tika versão 1.2

3.4.1 Apache Lucene

O Apache Lucene[5] é uma biblioteca de código aberto para contulas *full-text* feita em Java e tem como objetivo adicionar a funcionalidade de busca de forma fácil para uma aplicação ou site. A biblioteca é composta por 2 módulos principais: indexação e busca. A indexação processa os dados originais gerando uma estrutura de dados inter-relacionada eficiente para a pesquisa baseada em palavras-chave. A busca, por sua vez, consulta o índice pelas palavras digitadas em uma consulta e organiza os resultados pela similaridade do texto com a consulta.

O Lucene oferece um bom nível de abstração para um conjunto poderoso de técnicas baseadas no modelo Vetorial e Booleano. Isto permite que o desenvolvedor crie um engenho de busca sem necessariamente conhecer as rotinas e algoritmos de indexação, nem de busca, sendo necessário apenas utilizar a API disponibilizada.

[5]http://lucene.apache.org/ Acessado em 26/06/2013

O Lucene utiliza em seu índice a estrutura de dados chamada de índice invertido, onde cada termo adicionado possui uma referência para o arquivo que o contém, como mostrado na Figura 3.10. Os índices invertidos são estruturas compostas por duas partes: o vocabulário e as ocorrências. O vocabulário incorpora o conjunto de todas as palavras distintas existentes no arquivo. Para cada palavra do vocabulário são construídas listas que contêm as exatas posições nas quais aparecem dentro do texto. O conjunto de todas as listas é denominado de ocorrências. Em uma coleção de arquivos, para cada palavra existente é armazenado também o número do arquivo no qual ocorre. A busca num arquivo invertido se dá através da verificação das entradas do arquivo e a recuperação de todos os documentos que citam o termo usado [Baeza-Yates and Ribeiro-Neto, 1999]. Na Figura 3.10 temos o exemplo de um índice invertido onde temos como exemplo de vocabulários as palavras *hood*, *little*, entre outas. Através da utilização do índice invertido, é possível dizer que o vocabulário *hood* pode ser encontrado nos documentos com número 0 e 1. Já o vocabulário *little* pode ser encontrado nos documentos com número 0 e 2.

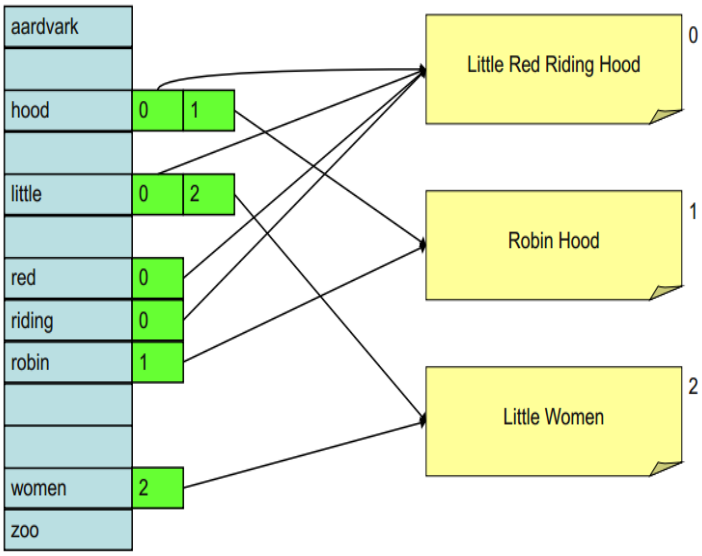

Figura 3.10 Índice Invertido no Apache Lucene [Hatcher and Gospodnetic, 2004].

Para cada documento presente no resultado de uma busca é atribuído uma pontuação que representa a similaridade de tal documento com a consulta. O cálculo dessa pontuação é feito baseando-se no modelo de recuperação de informação escolhido. No caso do Lucene, ele suporta alguns modelos, incluindo:

- Modelo Booleano;
- Modelo Espaço Vetorial;
- Modelo Probabilístico, como Okapi BM25 e DFR;
- Modelo baseado em Linguagem Natural;

A busca padrão no Lucene ocorre através da combinação de duas técnicas de Recuperação de Informação: Modelo Espaço Vetorial e modelo Booleano. Em geral, o modelo Booleano é utilizado primeiro para encontrar os documentos para os quais será calculado a pontuação de similaridade.

Este valor é calculado por meio da fórmula:

$$s(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{\text{t in d}} (tf(\text{t in d}) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d))$$

onde,

Fator	**Significado**
coord(q,d)	Quantos termos da consulta *q* aparecem no documento *d*
queryNorm(q)	Utilizado para fazer comparações entre consultas. Não afeta o ranqueamento
tf(t in d)	Frequência do termo *t* no documento d
idf(t)	Número de documentos que contêm o termo *t*.
t.getBoost()	Peso atribuído ao termo *t* no momento da consulta
norm(t,d)	Encapsula alguns outros pesos no momento da indexação, como fatores ligados ao documento, atributos de indexação e fatores ligados a similaridade

Em resumo, segundo o cálculo apresentado:

- Documentos contendo todos os termos são melhores;
- Correspondência entre os termos de busca e palavras raras nos metadados indexados é melhor do que com palavras comuns;
- Documentos curtos são melhores que documentos longos
- Documentos com muitas menções aos termos de buscas são bons;

Como o objetivo do Ustore é localizar e ordenar os resultados somente através da quantidade de termos encontrados no conteúdo indexado dos arquivos, este algoritmo de pontuação foi simplificado para levar em consideração apenas a quantidade de combinações entre os metadados indexados e os termos de busca.

Apesar da facilidade de uso, o Lucene contém apenas o núcleo do "motor"de busca. Por isto, ele não inclui um *Web crawler* ou um *parser* para diferentes formatos de arquivos. Este processamento, se necessário, deve ser feito pela aplicação ou por outra biblioteca e repassado ao Lucene por meio de instâncias da classe Document.

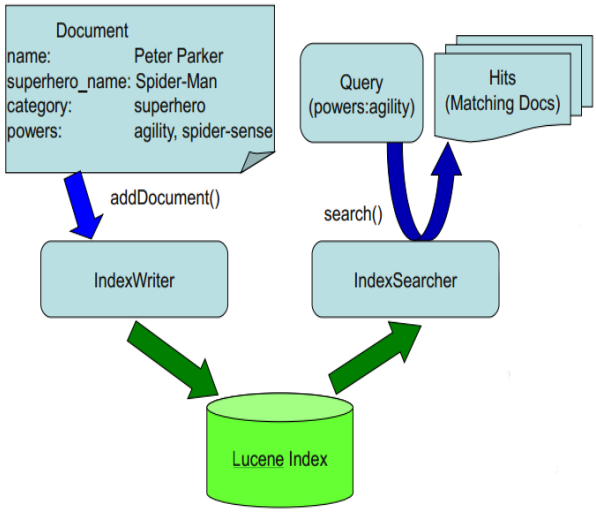

Figura 3.11 Arquitetura do Apache Lucene. [Hatcher and Gospodnetic, 2004]

3.4.2 Apache Tika

O Apache Tika é um subprojeto do Apache Lucene e é um kit de ferramentas para extrair metadados e conteúdo de diversos tipos de arquivos. A extração do conteúdo é feita por *parsers* específicos para cada tipo de arquivo. O Tika fornece uma API padrão e faz uso de bibliotecas de terceiros para fazer a extração do conteúdo, como a POI[6], usada para extrair o conteúdo de arquivos do Word, Excel, PowerPoint, e a PDFBox[7], usada para extração do conteúdo de arquivos PDF. O Apache Tika nativamente já vem com acesso aos *parsers* para trabalhar com arquivos com os formatos: HTML, XML, OLE2 e OOXML do Microsoft Office, OpenDocument Format, PDF, ePub, RTF, arquivos compactados e empacotados.

A arquitetura em alto nível do Apache Tika pode ser vista na Figura 3.12, onde é possível identificar uma camada chamada Tika Facade. Esta camada é responsável por receber o arquivo que será processado, identificar o tipo do mesmo, seu idioma e selecionar um *parser* adequado para extrair seus metadados e o conteúdo textual.

3.5 Sumário do Capítulo

Este capítulo apresentou uma abordagem para indexação e consultas *full-text* baseada em conteúdo para sistemas de armazenamento em nuvem. Além disso, foram apresentados os requisitos que foram desenvolvidos nesta abordagem, sua arquitetura e detalhes da implementação realizada. A abordagem proposta foi implementada no Ustore, um sistema real de armazenamento em nuvem, no qual a indexação dos arquivos acontece somente no clientes. Esta indexação extrai os metadados correspondentes e o conteúdo

[6]http://poi.apache.org/ Acessado em 23/05/2013
[7]http://pdfbox.apache.org/ Acessado em 23/05/2013

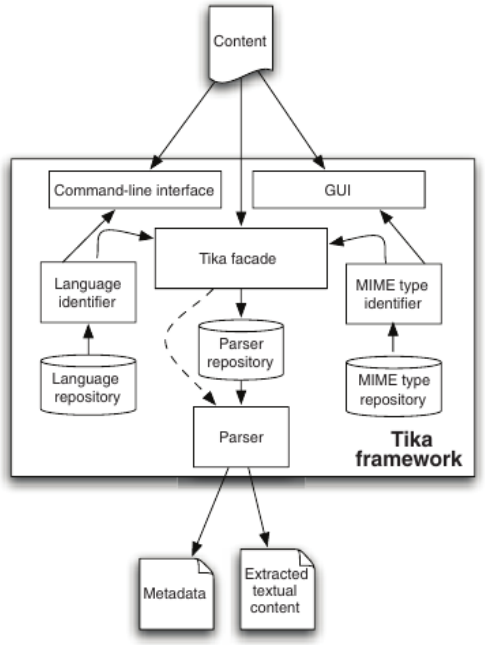

Figura 3.12 Arquitetura do Apache Tika. [Mattmann and Zitting, 2011]

dos arquivos. Estes metadados são indexados no próprio cliente e enviados para serem indexados nos Servidores de Busca. Com isso, o Ustore permite que as consultas *full-text* sejam realizadas nos Servidores, através da rede P2P ou apenas localmente

Conforme apresentado na Tabela 3.3, a maioria dos sistemas de armazenamento em nuvem hoje desenvolvidos oferecem opções simples para indexação e realização de consultas. Com a abordagem proposta, é possível oferecer opções avançadas de indexação e busca em sistemas de armazenamento em nuvem que armazenam apenas os *chunks* dos arquivos.

Tabela 3.3 Tabela comparando sistemas de armazenamento em nuvem.

Sistemas de Armazenamento em Nuvem	Forma de Armazenamento	Tipo de Consulta			
		Nome	Metadados	Palavras-Chave	Conteúdo
Amazon S3	Arquivo Inteiro	X	X		
Amazon S3 + Amazon Cloud Search	Arquivo Inteiro	X	X		
XtreemFS	Arquivo Inteiro	X	X		
BitTorrentSync	Chunks	X		X	
Box	-	X			X
Dropbox, SygarSync, Google Drive, Wuala	-	X			
Abordagem Proposta	Chunks	X	X		X

No próximo capítulo será apresentado um experimento e seus resultados, o qual foi realizado no Ustore com o objetivo de medir o desempenho da abordagem proposta.

4
Experimento

Este capitulo descreve o experimento e seus resultados realizado com o objetivo de avaliar o desempenho do engenho de busca em um sistema de armazenamento em nuvem, levando em consideração uma indexação baseada no nome do arquivo e a indexação baseada em conteúdo. Também avaliamos o impacto da inclusão da funcionalidade de indexação nos clientes do Ustore.

Como mostrado em [Manning et al., 2008], para realizar a avaliação de uma sistema desta natureza precisamos definir:

- Uma coleção de dados;
- As informações que serão solicitadas expressas em forma de consultas;
- Um conjunto de julgamentos de relevância;

Nós decidimos utilizar para validação um conjunto dados composto por arquivos pessoais e/ou profissionais. Por isso, tomamos a decisão de utilizar artigos acadêmicos indexados no engenho de busca Scopus[1]. Escolhemos como termo de busca a palavra-chave "*cloud computing*", por ser uma das palavras-chave deste trabalho, e realizamos o *download* dos 100 primeiros artigos ordenados pela quantidade de vezes que foram citados em outros artigos acadêmicos indexados. Geralmente, as avaliações de engenhos de busca utilizam mais arquivos, mas, devido à restrição de tempo, decidimos utilizar um conjunto de dados pequeno, pois, como será mostrado adiante, foi necessário realizar o *backup* de todo o conjunto de arquivos diversas vezes no Ustore para que fosse possível extrair os resultados para as métricas planejadas. Entretanto, mesmo com um conjunto de dados reduzido, é possível obter resultados satisfatórios.

Com o objetivo de comparar os resultados retornados no Scopus e no Ustore, o termo de busca escolhido para realizar as consultas no Ustore foi a palavra-chave "*cloud computing*", por ter sido a única utilizada no Scopus.

Para a realização dos testes é necessário que os arquivos tenham uma classificação que indique quais são os mais relevantes para cada consulta que será realizada no Ustore. Para fazer essa classificação no nosso conjunto de dados utilizamos a quantidade de vezes que os termos de busca são encontrados nos metadados e no conteúdo indexado, ou seja, quanto mais termos forem encontrados, mais relevante o arquivo é para a nossa consulta.

[1] www.scopus.com Acessado em: 13/07/2013

4.1 Ambiente de Teste

Os testes foram realizados utilizando 4 máquinas virtuais interligadas por uma rede Ethernet de 10/100 MBit/s, 2 delas (Win7.1 e Win7.2) executando Windows 7 com 1Gb de memória RAM e processador de 2.4 Ghz e 1 executando Linux Debian 5.0 com 1Gb de memória RAM e processador de 2.4 Ghz e 1 executando Linux CentOS com 1Gb de memória RAM e processador de 2.4 Ghz. O Servidor Ustore e o Super Peer foram hospedados na máquina CentOS. A máquina Debian hospedou 1 Servidor de Busca. Tanto a máquina Win7.1 quanto a Win7.2 receberam dois Clientes Ustore cada, conforme representado na Figura 4.1.

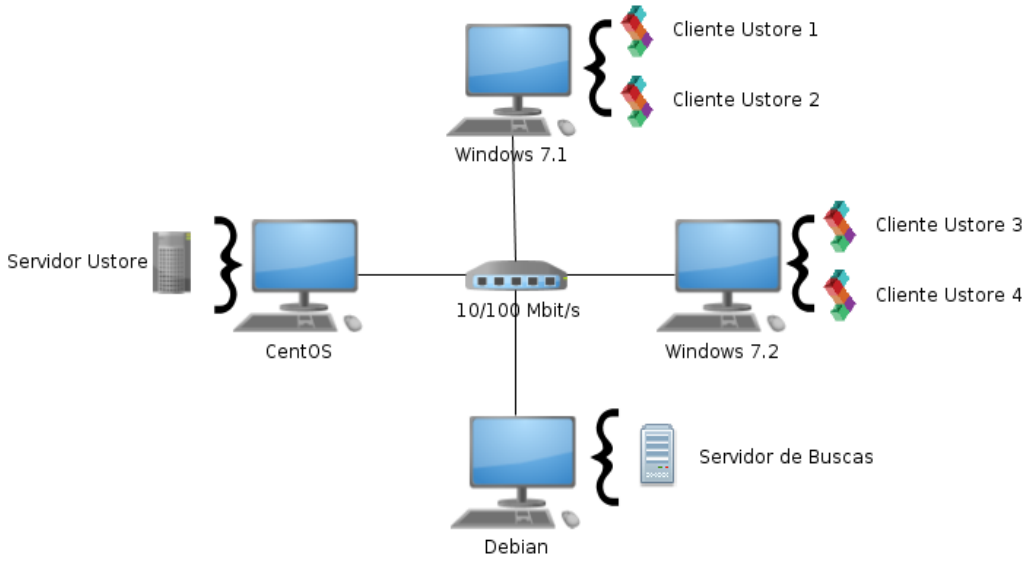

Figura 4.1 Representação do cenário de testes utilizado no experimento.

4.2 Metodologia de Avaliação

A metodologia de avaliação utilizada se baseia no método sistemático para avaliação de desempenho proposto em [Jain, 1991], no qual precisamos definir objetivos, métricas e fatores e níveis. O **objetivo** desta avaliação é avaliar o desempenho do engenho de busca em um sistema de armazenamento em nuvem levando em consideração a indexação baseada no nome do arquivo e a indexação baseada em conteúdo. E também avaliar qual o impacto da inclusão da funcionalidade de indexação baseada em conteúdo nos clientes do Ustore.

As **métricas** são os critérios para comparar o desempenho e estão sumarizadas na Tabela 4.1. As métricas escolhidas foram:

1. Precisão: A precisão representa a fração dos resultados retornados que são relevantes para a consulta. É dada através da proporção entre o número de documentos

Tabela 4.1 Métricas utilizadas na avaliação.

Métricas	Descrição
M_1: Precisão	Proporção entre o número de arquivos relevantes retornados e o número total de documentos recuperados
M_2: *Recall*	Proporção entre o número de documentos relevantes recuperados e o número total de documentos relevantes na base
M_3: *F-Measure*	Média harmônica ponderada da precisão e *recall*.
M_4: Tempo para realização de consultas	Tempo gasto para realizar uma consulta
M_5: Tempo para indexação	Tempo gasto para realizar a indexação de um arquivo

relevantes retornados e o número total de documentos recuperados e é representada pela fórmula [Baeza-Yates and Ribeiro-Neto, 1999]:

$$precisao = \frac{|\{documentos\ relevantes\} \cap \{documentos\ recuperados\}|}{|\{documentos\ recuperados\}|} \quad (4.1)$$

É representada por valores entre 0 (zero) e 1 (um) ou em termos de porcentagem. Quando mais próximo de 1 (um) ou 100% mais preciso é o sistema.

2. *Recall*: A taxa de *recall* representa a fração de documentos relevantes que foram retornados pela consulta. Ou seja, dado-se o conjunto de documentos recuperados, o *recall* é a proporção entre o número de documentos relevantes recuperados e o número total de documentos relevantes na base e é dado pela fórmula [Baeza-Yates and Ribeiro-Neto, 1999]:

$$recall = \frac{|\{documentos\ relevantes\} \cap \{documentos\ recuperados\}|}{|\{documentos\ relevantes\}|} \quad (4.2)$$

É representada por valores entre 0 (zero) e 1 (um) ou em termos de porcentagem, onde 0 (zero) representa que nenhum documento satisfaz a consulta e 1 (um) ou 100% representa todos os documentos indexados.

3. *F-Measure*: É a média harmônica ponderada da precisão e *recall*. Pode ser representada pela fórmula abaixo [Baeza-Yates and Ribeiro-Neto, 1999] :

$$F - Measure\ \alpha = \frac{(1+\alpha) * precisao * recall}{((\alpha * precisao) + recall} \quad (4.3)$$

Neste experimento as taxas de precisão e *recall* tem o mesmo fator de importância, portanto o valor de α é igual a 1 (um). Logo, esta função só retornará um valor no

intervalo entre zero e um. Quanto mais próximo de zero, menos relevantes são os documentos e quanto mais próximo de um, mais relevantes. A *F-Measure* só terá valores próximo de 1 (um) quando ambas as taxas de precisão e *recall* forem altas [Garcia et al., 2006; Durão et al., 2008].

4. Tempo para realização de consultas: Foram efetuadas 30 vezes a mesma consultas e medido o tempo gasto para realizá-las. Com os resultados obtidos será calculado um intervalo com 95% de confiança para representar estatisticamente o tempo necessário para realizar uma consulta. Um intervalo de confiança tem como objetivo mostrar o quanto determinada medida é precisa.

5. Tempo para indexação: Foram efetuadas 30 vezes a mesma consultas e medido o tempo gasto para realizá-las. Com os resultados obtidos será calculado um intervalo com 95% de confiança para representar estatisticamente o tempo necessário para realizar uma indexação.

Para efeito de comparação, esta avaliação considera como satisfatória taxas de precisão de 0.40 e *recall* de 0.45, resultados semelhantes encontrados em [Tang and Dwarkadas, 2004] e [Lu and Callan, 2003]. Utilizando estas taxas de precisão e *recall*, podemos calcular a taxa de *F-Measure* esperada. Esta ficou com o valor de 0.42. Para o tempo gasto para realizar as consultas, consideramos como satisfatório o tempo de 1000 milissegundos (ms), pois este foi o tempo alcançado em uma arquitetura *peer-to-peer* semelhante à utilizada pelo Ustore [Yang et al., 2006].

Tabela 4.2 Fatores e níveis utilizados para avaliar as métricas definidas.

Fatores	**Níveis**
Estratégia de indexação	Baseada no título dos arquivos e baseada no conteúdo arquivos
Estratégia de busca	Local, distribuída e Servidor de Buscas
Quantidade de resultados retornados	5, 10 e 15

O desempenho de um sistema é afetado por vários parâmetros, que podem ser divididos em variáveis ou não-variáveis. Os parâmetros que podem ser variados são chamados de **fatores** e os seus valores são chamados de **níveis**. Os fatores e níveis deste estudo estão listados na Tabela 4.2.

4.2.1 Cenários para Avaliação

A avaliação de desempenho foi dividida em quatro cenários. A divisão em cenários visa facilitar o entendimento do que está sendo avaliado e também torna mais fácil variar os fatores. Os cenário foram montados da seguinte forma:

- **Cenário 1:** No primeiro cenário o objetivo foi avaliar a busca local. Para isso, a indexação de títulos e conteúdo foi feita localmente, ou seja, todo o índice foi

gerado em cada cliente que efetuou *backup*. Neste cenário foram avaliadas as métricas M_1 (precisão), M_2 (*recall*), M_3 (*F-Measure*) e M_4 (tempo de consulta) variando a estratégia de indexação utilizada e a quantidade de retornos.

- **Cenário 2:** No segundo cenário o objetivo foi avaliar a busca distribuída. Para isso, a indexação de títulos e conteúdo foi feita localmente, ou seja, todo o índice foi gerado em cada cliente que efetuou *backup*. Com isso é possível enviar uma requisição para que os outros clientes façam uma consulta aos seus respectivos índices. Neste cenário foram avaliadas as métricas M_1 (precisão), M_2 (*recall*), M_3 (*F-Measure*) e M_4 (tempo de consulta) variando a estratégia de indexação utilizada e a quantidade de retornos.

- **Cenário 3:** No terceiro cenário o objetivo foi avaliar a busca no Servidor de Buscas. Para que isso fosse possível, cada cliente, ao finalizar um *backup*, enviou os metadados de cada arquivos para o Servidor de Buscas, que os indexou em um único índice. Neste cenário foram avaliadas as métricas M_1 (precisão), M_2 (*recall*), M_3 (*F-Measure*) e M_4 (tempo de consulta) variando a estratégia de indexação utilizada e a quantidade de retornos.

- **Cenário 4:** O último cenário tem como objetivo analisar o impacto da inclusão da funcionalidade de indexação e buscas no Ustore. Neste cenário foi analisada a métrica M_5 (tempo de *backup*). Para isto, foi realizado o cálculo estatístico do intervalo de confiança com 95% de confiança e taxa de erro de 5% para o tempo de *backup* no Ustore em três situações: sem indexação, com indexação local e com indexação local e no servidor.

4.3 Resultados

O experimento foi executados nos quatro cenário de avaliação e a seguir são apresentados os resultados obtidos.

4.3.1 Resultados no Cenário I

No primeiro cenário utilizamos os quatro clientes Ustore (U1, U2, U3 e U4) para fazer *backup* de todos os arquivos, os quais foram distribuídos aleatoriamente entre os clientes de forma que todos ficassem com o mesmo número de arquivos (25 arquivos para cada um). Neste cenário realizamos a indexação e busca locais, ou seja, foram gerados apenas índices em cada cliente que efetuou *backup* e utilizamos somente estes índices para responder às consultas.

Tabela 4.3 Tabela com tamanho dos índices gerados em cada cliente Ustore.

Estratégia de Indexação	U1	U2	U3	U4
Título	20.5 Kb	20.7 Kb	21.5 Kb	21.8 Kb
Conteúdo	2 Mb	2 Mb	2.6 Mb	2 Mb

A Tabela 4.3 mostra o tamanho dos índices gerados em cada cliente Ustore. Nota-se que o índice gerado com a indexação baseada em conteúdo é maior do que o índice gerado com a indexação baseada em título. Isso se deve pela grande quantidade de informação armazenada no atributo *CONTENT* dos metadados de cada arquivos, descrito na Seção 3.3.2 do Capítulo 3.

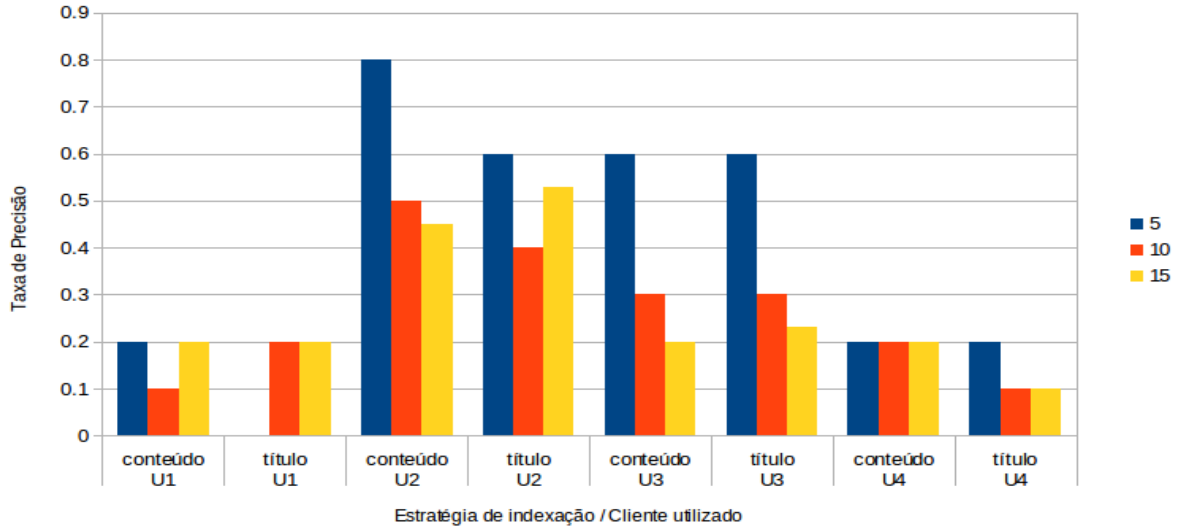

Figura 4.2 Gráfico comparando a precisão entre quatro clientes do Ustore variando a quantidade de retornos e o tipo de indexação.

A Figura 4.2 mostra a precisão alcançada utilizando os quatro clientes Ustore utilizando indexação por conteúdo e por título dos arquivos e variando a quantidade de retornos (5, 10 e 15) e utilizando para consultas apenas o índice local em cada cliente. Pelo gráfico percebemos que a maior parte dos arquivos relevantes ficaram nos clientes U2 e U3, pois estes foram os que obtiveram as maiores taxa de precisão. Como percebemos pelo gráfico, na maior parte dos testes realizados, as taxas de precisão quando utilizamos a indexação baseada em conteúdo são mais altas do que quando utilizamos a indexação baseada em título, por exemplo, quando consideramos apenas o cliente U2 temos taxas de precisão de 0.8, 0.5 e 0.45 para buscas baseadas em conteúdo e 0.6, 0.4 e 0.53 para buscas baseadas em título, com número de retornos de 5, 10 e 15, respectivamente. Com isso, temos que, para 2 das 3 buscas realizadas, as taxas de precisão são superiores para o primeiro caso. Devido à distribuição aleatória dos arquivos, alguns clientes indexaram poucos arquivos relevantes e, por isso, possuem taxas de precisão baixas. Se considerarmos a média aritmética das taxas de precisão alcançadas para um retorno de 5 resultados, conseguimos alcançar uma taxa satisfatória de 0.45. Já, ao considerarmos as outras taxas de retornos, as médias obtidas mostram que precisamos melhorar a qualidade dos resultados, já que essas não alcançaram o valor de referência obtidos por outros autores.

A Figura 4.3 mostra as taxas de *recall* alcançadas com os quatro clientes do Ustore utilizando indexação por conteúdo e por título dos arquivos e variando a quantidade de

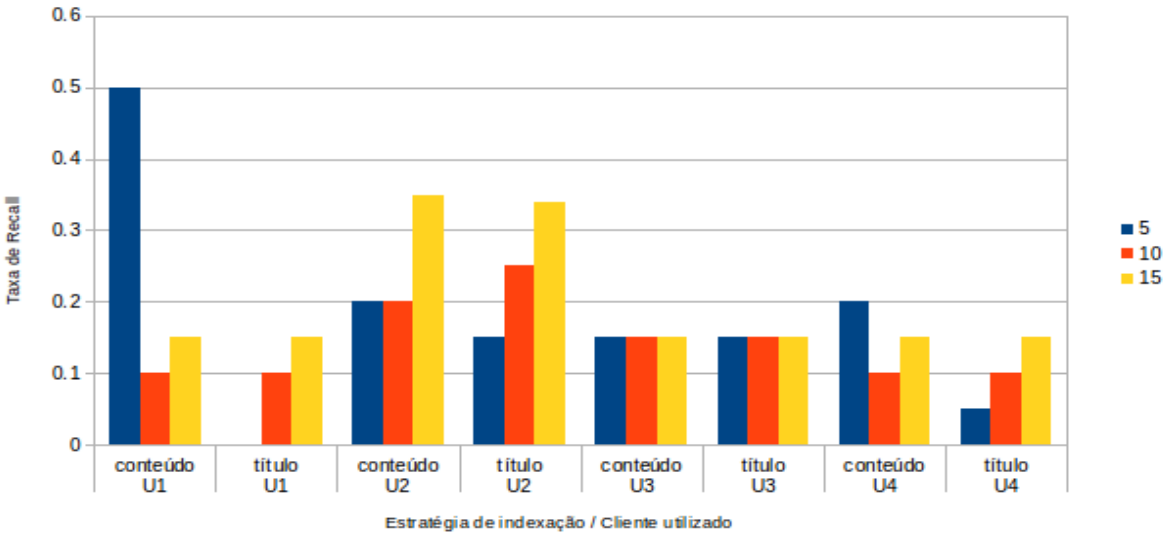

Figura 4.3 Gráfico comparando o *recall* entre quatro clientes do Ustore variando a quantidade de retornos e o tipo de indexação.

retornos (5, 10 e 15). Assim como na precisão, as taxas obtidas com o *recall* quando utilizamos a indexação por conteúdo são superiores às taxas obtidas quando utilizamos a indexação por título. A análise do gráfico gerado permite observar que quando consideramos apenas os 5 (cinco) primeiros resultados, o cliente U1 possui uma taxa de *recall* significante (0.5) para a busca baseada no conteúdo. Considerando os 15 (quinze) primeiros resultados, o cliente U2 possui as taxas mais altas de *recall*: 0.35 e 0.34, respectivamente para buscas baseada em conteúdo e título. Analisando o gráfico de forma geral, percebe-se que as taxas de *recall* não são significativamente altas, ou seja, muitos resultados relevantes não foram recuperados na consulta As médias aritméticas relativas às taxas de retorno mostram que melhorias urgentes são necessárias para melhorar a qualidade do *recall* do Ustore.

Com as taxas de precisão e *recall* definidas podemos calcular a *F-Measure* para cada cliente Ustore apresentado. Os resultados estão na Tabela 4.4.

Tabela 4.4 Tabela com os valores da *F-Measure*.

Retornos	U1		U2		U3		U4	
	Conteúdo	Título	Conteúdo	Título	Conteúdo	Título	Conteúdo	Título
5	0.29	0.00	0.32	0.24	0.24	0.24	0.20	0.08
10	0.10	0.13	0.29	0.31	0.20	0.20	0.13	0.10
15	0.17	0.17	0.39	0.41	0.17	0.18	0.17	0.12

Analisando os dados apresentados na Tabela 4.4 percebemos que não alcançamos valores muitos altos (próximos a 1), pois a *F-Measure* é derivida das taxas de precisão e *recall* e, como apresentado anteriormente, as taxas de *recall* alcançadas não foram altas.

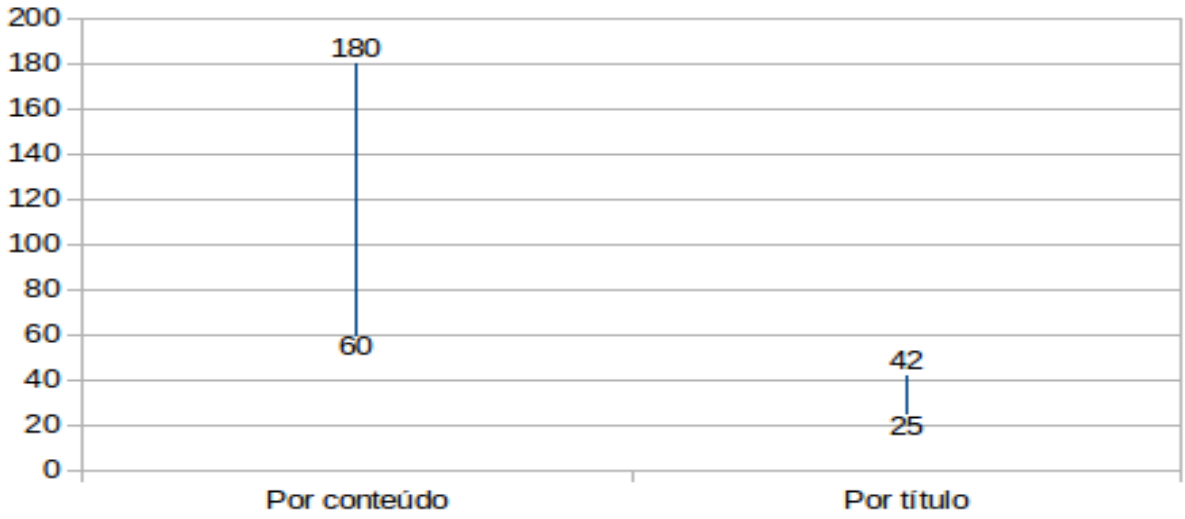

Figura 4.4 Gráfico comparando o tempo de consulta local do Ustore variando a quantidade de retornos e o tipo de indexação.

A Figura 4.4 mostra a medição de tempo para realizar 30 consultas locais no Ustore utilizando indexação baseada em título e em conteúdo. Nela podemos perceber que, devido à maior quantidade de informação gerada pela indexação baseada em conteúdo, as consultas realizadas neste índice são mais demoradas do que aquelas realizadas no índice com indexação baseada apenas no título. Foi realizada uma análise estatística dos resultados obtidos e ela mostra que em 95% dos casos uma consulta no índice gerado com o conteúdo dos arquivos demora entre 60ms e 180ms; já uma consulta no índice gerado apenas com o título dos arquivos demora entre 25ms e 42ms. Como descrito anteriormente, consideramos o tempo de 1000ms como satisfatório para a realização de uma consulta, logo os tempos obtido para a consulta baseada em conteúdo são satisfatórios.

4.3.2 Resultados no Cenário II

No segundo cenário foram utilizados os mesmo clientes Ustore anteriores com os mesmo arquivos indexados. A única diferença significativa entre o primeiro e segundo cenário, foi a estratégia de busca adotada. Enquanto no primeiro cenário a busca foi realizada localmente, neste a busca foi realizada utilizando os outros clientes ativos, ou seja, a busca foi realizada através da rede JXTA formada entre os clientes.

A Figura 4.5 mostra a precisão alcançada utilizando os quatro clientes Ustore utilizando indexação por conteúdo e por título dos arquivos e variando a quantidade de retornos (5, 10 e 15). A análise do gráfico gerado permite dizer que os clientes U1, U2 e U4 conseguem taxas satisfatórias de precisão, 0.5 e 0.4 e 0.4, respectivamente, com taxas de retornos variáveis. Considerando todo os clientes, temos que as taxas de precisão

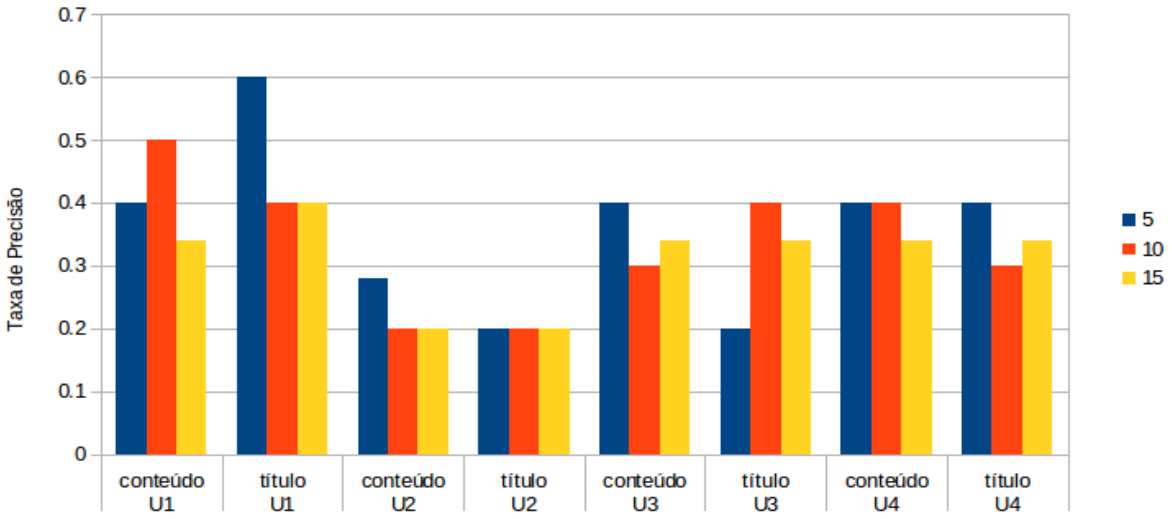

Figura 4.5 Gráfico comparando a precisão entre quatro clientes do Ustore variando a quantidade de retornos e o tipo de indexação e fazendo consulta aos outros clientes.

são semelhantes quando comparamos as buscas realizadas baseadas no conteúdo e título. A análise das médias aritméticas para cada variação da quantidade de retornos mostra que melhorias precisam ser aplicadas para aumentar a precisão do Ustore nas consultas através da rede, pois a média mais alta que obtemos foi de 0.37, enquanto a média de referência é de 0.45.

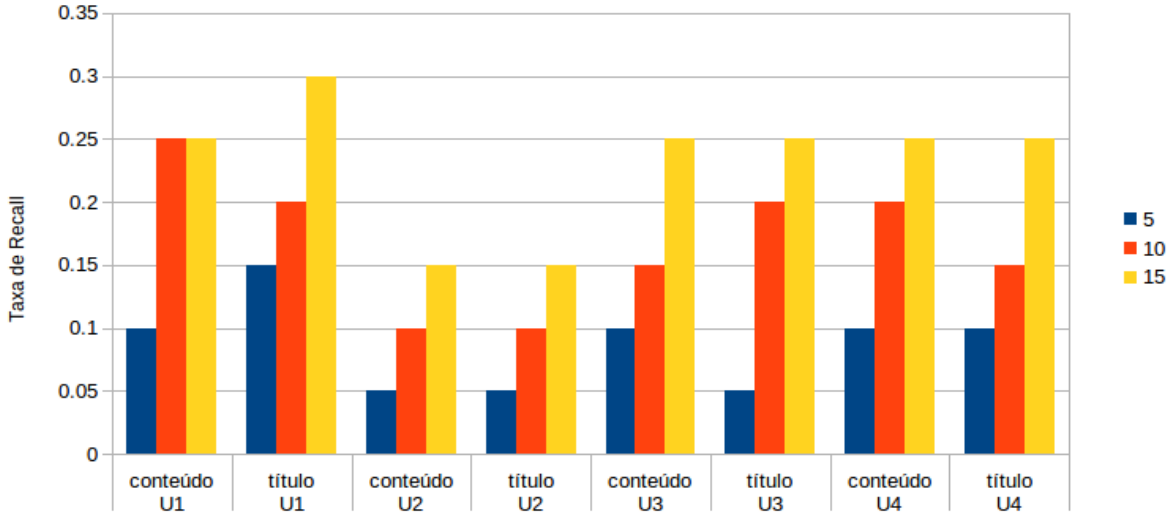

Figura 4.6 Gráfico comparando o *recall* de quatro clientes do Ustore variando a quantidade de retornos e o tipo de indexação e fazendo consulta aos outros clientes.

A Figura 4.6 mostra a taxa de *recall* dos clientes Ustore. O gráfico mostra que os clientes Ustore conseguem taxas de *recall* semelhantes, independente da estratégia de indexação utilizada (conteúdo ou título). Para este cenário, foram alcançadas taxas de *recall* muito baixas, assim como no Cenário I. Em alguns casos, como no cliente U2, as taxas alcançam uma média de apenas 0.1. As médias aritméticas obtidas para cada variação da quantidade de retornos mostram que melhoria urgentes precisam ser realizadas para aumentar a qualidade do *recall* das consultas através da rede. A média mais alta alcançada neste cenário foi de 0.22.

Com as taxas de precisão e *recall* definidas podemos calcular a média F para cada cliente Ustore. Os resultados estão na Tabela 4.5.

Tabela 4.5 Tabela com os valores da *F-Measure*.

Retornos	U1		U2		U3		U4	
	Conteúdo	Título	Conteúdo	Título	Conteúdo	Título	Conteúdo	Título
5	0.16	0.24	0.08	0.08	0.16	0.08	0.16	0.16
10	0.33	0.27	0.13	0.13	0.20	0.27	0.27	0.20
15	0.29	0.34	0.17	0.17	0.29	0.29	0.29	0.29

Analisando os dados apresentados na Tabela 4.5 percebemos que não alcançamos valores muitos altos (próximos a 1), pois a *F-Measure* é derivida das taxas de precisão e *recall* e, como apresentado anteriormente, as taxas de *recall* alcançadas não foram altas.

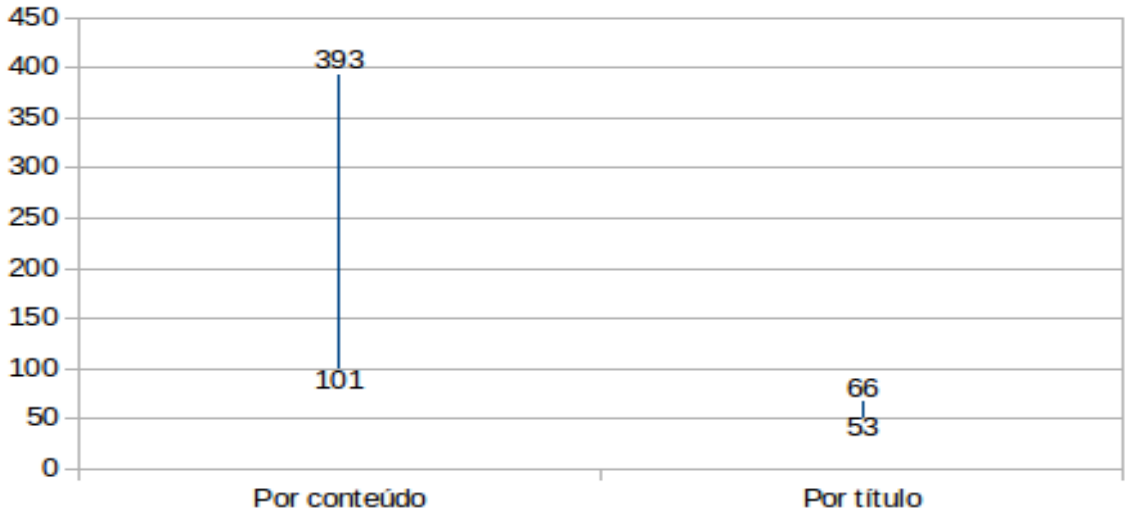

Figura 4.7 Gráfico comparando o tempo de consulta aos outros clientes do Ustore variando a quantidade de retornos e o tipo de indexação.

A Figura 4.7 mostra o tempo gasto para a execução de 30 consultas no Ustore utilizando os outros clientes ativos para respondê-las. A análise do gráfico gerado

permite observar que, na maioria dos casos, a busca baseada em conteúdo demora mais para executar do que a busca por título. Isso se justifica pelo fato da indexação do conteúdo gerar um índice de busca maior, como mostrado anteriormente. Há consultas que demoraram muito mais que outras, por exemplo, a primeira e quarta consulta por conteúdo. Os testes foram executados durante a execução normal do Ustore e o mesmo continua enviando outros tipos de mensagens, como: disponibilidade, replicação de *chunks* e isto pode ter causado esta elevação no tempo necessário para concluir uma consulta. A análise estatística nos permite calcular, com 95% de confiança, que as consultas através da rede baseadas em conteúdo serão realizadas no intervalo de 101ms e 393ms. Já as consultas baseadas em título serão realizadas no intervalo de 53ms e 66ms. Os resultados obtidos no intervalo de confiança para realização de uma consulta se mostraram satisfatórios, pois, como descrito anteriormente, consideramos o tempo de 1000ms como satisfatório para a realização de uma consulta.

4.3.3 Resultados no Cenário III

O terceiro cenário leva em consideração somente a indexação e consultas realizadas no Servidor de Buscas. Neste cenário o Servidor de Busca contém um índice de busca gerado pela indexação de todo o conjunto de dados de 100 arquivos com extensão PDF, com o tamanho total de 4.6 MB para o índice com o conteúdo dos arquivos e 76,7 KB para o índice gerado pela indexação baseada apenas no título dos arquivos.

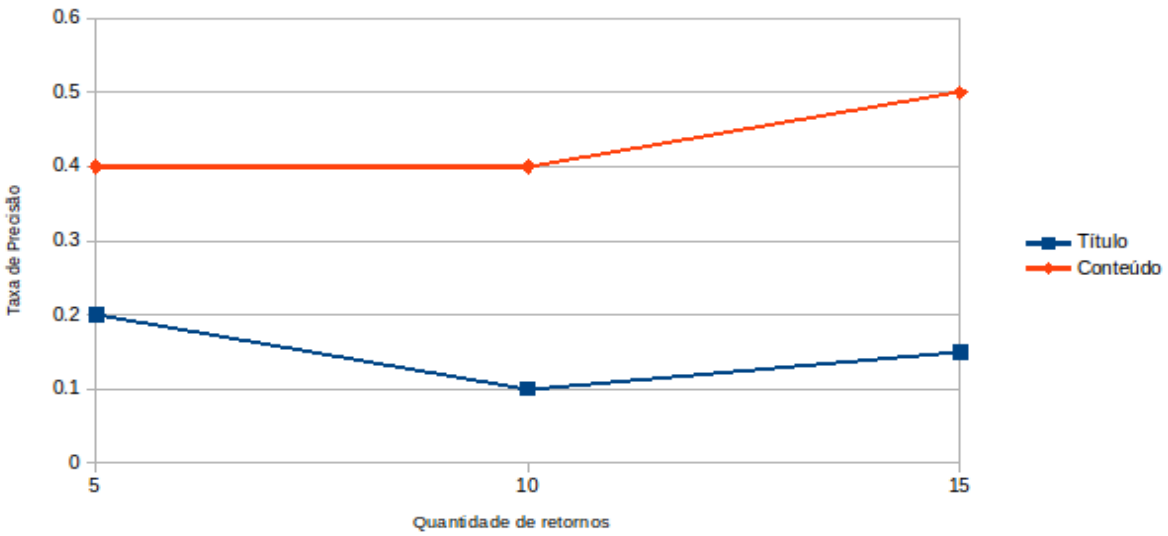

Figura 4.8 Gráfico comparando a precisão do Ustore utilizando o Servidor de Buscas variando a quantidade de retornos e o tipo de indexação.

A Figura 4.8 mostra o resultado da consulta pela palavra-chave "*cloud computing*" variando a quantidade de retornos e o tipo de indexação usada no Ustore. Nela podemos ver que a indexação por conteúdo possui uma precisão melhor do que a indexação

por título. Por exemplo, quando analisamos o gráfico com uma taxa de retorno de 10 resultados, obtemos 0.40 de precisão com a indexação baseada em conteúdo, enquanto obtemos somente 10% de precisão com a indexação por título. Já com um retorno de 15 resultados, temos 0.5 e 0.15, respectivamente.

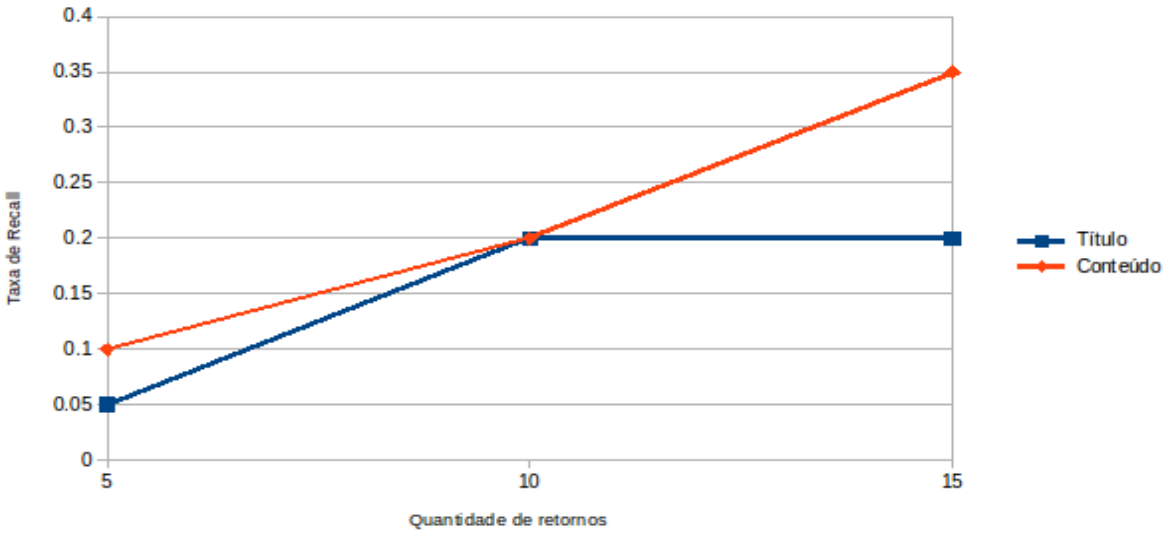

Figura 4.9 Gráfico comparando o *recall* do Ustore utilizando o Servidor de Buscas variando a quantidade de retornos e o tipo de indexação.

A Figura 4.9 mostra a taxa de *recall* alcançada. Nele notamos que a taxa de *recall* cresce quando aumentamos a quantidade de retornos para uma busca baseada em conteúdo. Já para uma busca baseada no título, esta taxa se manteve constante para 10 e 15 resultados. Neste cenário também obtivemos taxa consideravelmente baixas, o que indica que muito arquivos relevantes não foram retornados nas consultas.

Unificamos os dois resultados, de forma que possamos comparar as taxas obtidas levando em consideração as duas métricas, simultaneamente. Na Figura 4.10 temos o gráfico gerado e nele podemos notar que com uma taxa de retorno de 15 resultados é possível obter uma taxa satisfatória de precisão (0.50) juntamente com uma taxa razoável de *recall* (0.35) para a busca baseada no conteúdo. Já para as consultas baseadas no título dos arquivos, um retorno de 15 resultados alcança uma taxa de 0.15 de precisão e 0.2 de *recall*, ambas inferiores às mesmas consultas baseada em conteúdo.

Com os valores de precisão e *recall* definidos, é possível calcular o valor da média F. Os valores encontrados são mostrados na Tabela 4.6.

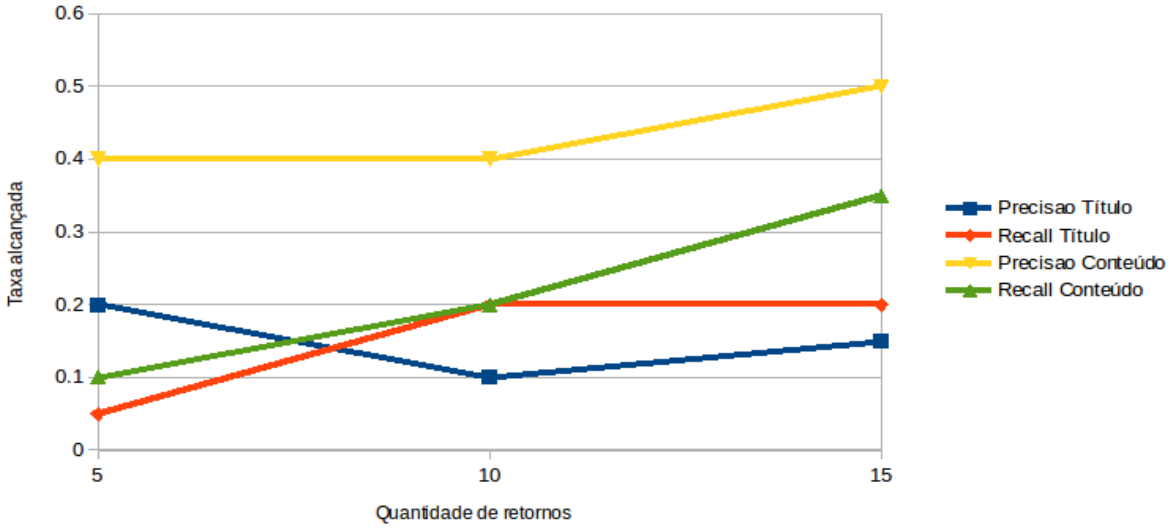

Figura 4.10 Gráfico comparando as taxas de precisão e *recall* do Ustore variando a quantidade de retornos e o tipo de indexação.

Tabela 4.6 Tabela com os valores da *F-Measure*.

Qnt. Retornos	Título	Conteúdo
5	0.08	0.16
10	0.13	0.27
15	0.17	0.41

Como mostrado anteriormente, a *F-measure* tem valores baixos para a razão entre precisão e *recall* devido, principalmente, às baixas taxas de *recall*. Entretanto, se considerarmos somente a taxa de *F-measure* alcançada quando utilizamos a consulta baseada em conteúdo e com uma taxa de retorno de 15 resultados temos um valor de 0.41, muito próximo ao valor inicialmente estabelecido de 0.42.

Na Figura 4.11 temos o tempo gasto para realizar 30 consultas pela palavra-chave "*cloud computing*" no Ustore utilizando indexação baseada em título e em conteúdo. O tempo gasto para efetuar as consultas no segundo caso é superior ao primeiro devido ao tamanho do índice que foi gerado, pois, como mostrado anteriomente, este é, aproximadamete, 60% maior. Utilizando o teste estatístico do intervalo de confiança, é possível dizer que em 95% do casos a busca baseada no título demorará entre 47ms e 58ms. Já a busca baseada no conteúdo demorará entre 461ms e 519ms. Como descrito anteriormente, consideramos o tempo de 1000ms como satisfatório para a realização de uma consulta, logo os tempos obtidos pelo intervalo de confiança para a consulta baseada em conteúdo são satisfatórios.

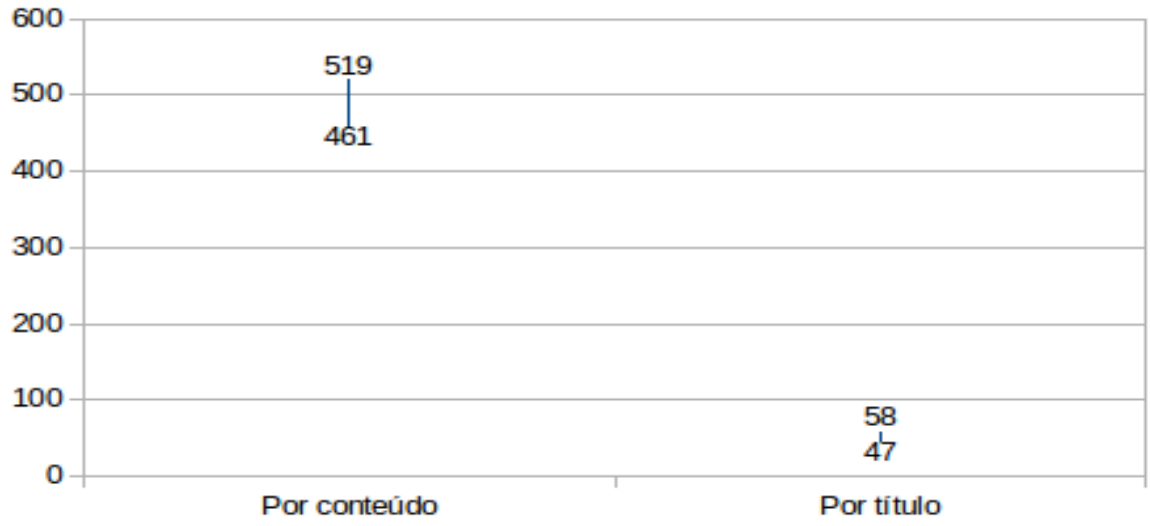

Figura 4.11 Gráfico comparando o tempo gasto para realizar consultas no Ustore utilizando indexação baseada em título e conteúdo.

4.3.4 Resultados no Cenário IV

O quarto cenário tem como objetivo avaliar o impacto da inclusão da funcionalidade de indexação no sistema Ustore. Para isso foram realizados 30 *backups* de um mesmo arquivo com extensão pdf e com tamanho de 12 Mb para avaliar o aumento no tempo gasto para a conclusão da tarefa para as três situações possíveis: sem indexação, indexação local e indexação no servidor. A Figura 4.12 mostra o resultado dos *backups* realizados.

A análise através do cálculo estatístico do intervalo de confiança nos mostra que em 95% dos casos o *backup* sem a realização de nenhuma indexação gastará entre 2647ms e 3922ms. O *backup* somente com indexação local gastará entre 2989ms e 4087ms. Já o *backup* com indexação local e no servidor gastará entre 5714ms e 7441ms.

4.4 Discussão e Considerações Finais

Nesta Seção nós discutimos os resultados obtidos e sua avaliação, restrições e oportunidades. Nós também discutimos possíve ameaças à validade de nosso experimento.

4.4.1 Discussão dos Resultados

Os experimentos mostram que a indexação baseada em conteúdo faz com que o índice tenha um crescimento significativo, como mostrado na Tabela 4.3. Este aumento é de cerca de 100% devido à grande quantidade de informação adicionada no atributo *CONTENT* dos metadados. Este aumento faz com que as consultas demorem mais tempo para serem executadas, como mostrado nas Figuras 4.4, 4.7 e 4.11. A Figura 4.4 apresenta

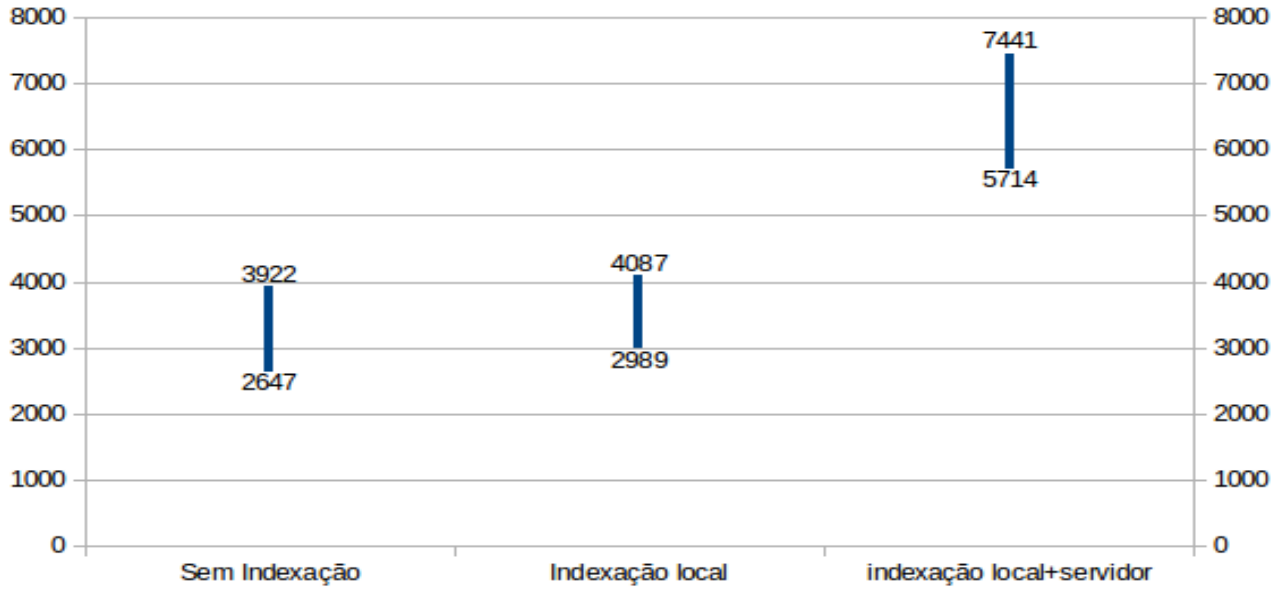

Figura 4.12 Gráfico comparando o tempo gasto realizar *backup* com e sem a indexação baseada em conteúdo gerando um índice local e outro no servidor.

uma diferença entre as consultas de 35ms para as consultas mais rápidas e 138ms para as consultas mais demoradas. A Figura 4.7 apresenta uma diferença de 48ms para consultas mais rápidas e 327ms para consultas as consultas mais demoradas. A Figura 4.11 mostra uma diferença de 414ms para consultas mais rápidas e 461ms para consultas as consultas mais demoradas. Mesmo com uma elevada diferença entre os tempos de consulta, estas continuaram sendo respondidas em um intervalo de tempo abaixo de 500ms. Como o tempo de referência utilizado foi de 1000ms, temos que o Ustore continua respondendo as consultas em um tempo consideravelmente baixo.

A análise do impacto da implantação da funcionalidade de indexação baseada em conteúdo no Ustore no Cenário IV mostra que a utilização somente da indexação local tem um impacto pequeno de aproximadamente 300ms para a indexação mais rápida e 160ms para a indexação mais demorada. Já com a utilização da indexação local e no servidor, o impacto é maior: cerca de 2700ms para a indexação mais rápida e 3300ms para a indexação mais demorada. Este aumento se dá pela necessidade de enviar uma mensagem assíncrona para o servidor com os metadados para serem indexados.

Os experimentos também mostram que apesar do aumento significativo no tamanho do índice gerado, no tempo de resposta da consulta e no tempo de indexação, a precisão, *recall* e *F-measure* do sistema melhoraram utilizando a abordagem proposta. A precisão no cenário 1 atingiu uma média de 0,33 quando consideramos o conteúdo dos arquivos e 0,28 quando consideramos somente o título. A taxa de *recall* alcançou 0,2 utilizando o conteúdo dos arquivos e 0,14 utilizando somente o título. No cenário 2, as taxas de precisão e *recall* são praticamente as mesmas para as duas situações (utilizando conteúdo e

título). No cenário 3, a taxa de precisão foi significativamente maior quando consideramos o conteúdo dos arquivos e atingiu uma média de 0,43 (superando o nível estabelecido como ideal). Já quando consideramos somente o título dos arquivos, obtivemos uma média de precisão de 0,15. A taxa de *recall* também foi maior quando consideramos o conteúdo: 0,21 contra 0,15, quando consideramos somente o título.

É importante ressaltar que os valores obtidos nestas métricas alcançaram os valores de referências inicialmente estabelecidos apenas em poucos casos, o que mostra que melhorias são necessárias.

A utilização de um índice centralizado no Servidor de Buscas permite que o usuário tenha acesso ao índice completo gerado no Ustore e, com isso, permite que a consulta leve em consideração todos os arquivos públicos de todos os usuários. Entretanto, como já discutido, este modelo tem diversas desvantagens como: ponto único de falha na rede, possibilidade de sobrecarga, entre outros. Com a utilização da abordagem proposta para armazenamento do índice no Servidor de Buscas e também localmente juntamente com a indexação baseada no conteúdo do arquivos, é possível minimizar estes problemas e, em caso de falha do servidor, é possível obter resultados satisfatórios, como mostrado pelo experimento executado, utilizando a busca através da rede ou até mesmo somente a busca local. Isto permite que o usuário obtenha retorno para suas consultas independente de problemas que possam acontecer com o Servidor.

4.4.2 Problemas Encontrados

Analisando os resultado finais é possível observar que a busca baseada em conteúdo traz resultados satisfatórios. Entretanto, a avaliação também revela alguns problemas:

- **Taxas baixas de precisão e *recall*:** Na maioria dos casos analisados as taxas de precisão e *recall* foram superiores para a indexação baseada em conteúdo. Entretanto, os resultados alcançados, na maioria dos casos, não alcançou o nível satisfatório estabelecido. Isto pode ter acontecido devido à classificação de relevância adotada na abordagem, já que consideramos que o documento mais relevante é aquele no qual podemos encontrar, em seus metadados e conteúdo, a maior quantidade de repetições dos termos buscados. Apesar do requisito não-funcional "Alta precisão e *recall*" ter sido "Parcialmente Atendido" (Seção 3.2.2), melhorias ainda podem ser realizadas.

- **Crescimento do índice de busca:** O índice gerado para buscas baseadas no conteúdo dos arquivos tem um aumento de tamanho considerável em relação ao índice para buscas baseadas no título dos arquivos. Este aumento aconteceu devido a decisão de manter no índice todo o conteúdo armazenado para ser utilizado futuramente, por exemplo, para calcular a similaridade entre arquivos e poder utilizar uma abordagem para recomendação de informação. O conteúdo armazenado não é necessário para a realização das consultas, pois este já foi indexado pelo Lucene e é mantido em uma estrutura de índice invertido, como descrito na Seção 3.4.1.

4.4.3 Possíveis Ameaças à Validade

O experimento executado tem algumas limitações, as quais devem ser levadas em consideração em uma possível replicação do mesmo, como:

- **Conjunto de dados pequeno:** Devido a restrições de tempo, nosso experimento foi realizado com um conjunto pequeno de dados. Deve-se considerar a utilização de uma quantidade maior de arquivos ou a utilização de um conjunto de dados do TREC[2].
- **Apenas uma consulta foi realizada:** Com a massa de dados indexada, apenas uma consulta foi realizada. Esta utilizou a palavra-chave "*cloud computing*". Deve-se considerar aumentar a quantidade de consultas realizadas e quantidade de palavras-chave utilizadas.
- **Ambiente totalmente distribuído:** No experimento realizado, os clientes, servidor e servidor de busca do Ustore foram executados em máquinas virtual em um mesmo servidor físico. Deve-se considerar a utilização de um ambiente totalmente distribuído, de forma a simular um ambiente real de computação em nuvem.
- **Métricas de avaliação:** Poucas métricas foram utilizas na avaliação de desempenho. Pode-se utilizar outras métricas relacionadas com recuperação de informação, como: curva de precisão-*recall*, bem como, computação em nuvem, como: *throughput* dos Servidores de Busca.

4.5 Sumário do Capítulo

Neste capítulo foi apresentado o experimento realizado no Ustore com o objetivo de avaliar a abordagem proposta. A avaliação foi feita para analisar o impacto de sua adoção no Ustore, bem como avaliar os níveis de precisão, *recall* e *f-measure* obtidos, os quais utilizados para analisar se houve aumento na quantidade de resultados relevantes retornados para o usuário. A abordagem de indexação e buscas baseadas em conteúdo foi comparada com a abordagem de busca baseada em título, utilizada pelo maior parte dos sistemas de armazenados existentes. O resultados mostraram que a abordagem proposta consegue aumentar, em alguns casos, a quantidade de resultados relevantes retornados e que se mostra viável para ser utilizada em sistemas reais de armazenamento em nuvem.

No próximo capítulo serão apresentadas as conclusões deste trabalho, as contribuições alcançadas e alguns possíveis trabalhos futuros.

[2]http://trec.nist.gov/ Acessado em 12/08/2013

5
Conclusão e Trabalhos Futuros

A utilização de sistemas para armazenamento de dados em nuvem se tornarão ainda mais necessários com o enorme volume de dados que será gerado nos próximos anos. Já existem diversos sistemas com este propósito, conforme descrito em capítulos anteriores. Entretanto, os sistemas de armazenamento em nuvem disponíveis não trazem recursos avançados para recuperação de informações, como buscas *full-text* e buscas baseadas no conteúdo. Neste contexto, foi proposto e desenvolvido uma abordagem com este objetivo, a qual se baseou nos requisitos definido na Seção 3.2 e nós a avaliamos em um sistema de armazenamento em nuvem real no Capítulo 4.

Com nossa abordagem é possível indexar e realizar consultas *full-text* baseadas em conteúdo em sistemas de armazenamento em nuvem. A implementação da abordagem proposta realizada no Ustore mostra que a abordagem proposta é viável e prática para ser utilizada em um ambiente real, como evidenciado pelo experimento realizado. O experimento mostrou que a utilização da abordagem permitiu que o conteúdo fosse indexado juntamente com os metadados do arquivos e que as métricas analisadas (precisão, *recall* e *f-measure*) indicam uma melhoria na qualidade dos resultados, em comparação com a busca baseada em título, a qual é utilizada pela maioria dos sistemas de armazenamento em nuvem. Os outros requisitos propostos no Capítulo 3 também foram alcançados, alguns de forma parcial e outros totalmente, como discuto anteriormente.

5.1 Contribuições

As principais contribuições deste trabalho foram: (a) uma abordagem para permitir indexação e consultas *full-text* baseadas em conteúdo, (b) a implementação da abordagem no Ustore e (c) um estudo experimental para medir o desempenho da abordagem. Estas contribuições estão detalhadas abaixo.

- **Uma abordagem para permitir indexação e consultas *full-text* baseadas em conteúdo** para sistemas de armazenamento em nuvem foi desenvolvida. Seus requisitos, arquitetura, implementação foram apresentados ao longo desta dissertação.

- **A implementação da abordagem no Ustore**, o que permitiu avaliar e demonstrar sua efetividade em um sistema real de armazenamento em nuvem.

- **Um estudo experimental** foi realizado com o objetivo de medir o desempenho da abordagem *full-text* em comparação com a abordagem baseada em títulos dos arquivos, a qual é utilizada por muitos sistemas de armazenamento. Este experimento foi baseado no processo para avaliação de desempenho proposto em [Jain, 1991].

5.2 Trabalhos Futuros

Devido a restrições de tempo, esta dissertação aborda alguns problemas, mas alguns outros ainda estão abertos. Além disso, ainda é possível melhorar alguns pontos nesta dissertação e evoluir a proposta apresentada. Estes são alguns pontos que podem ser investigados como trabalhos futuros:

- **Estender o experimento executado:** O experimento pode ser estendido de forma que seja validada toda a abordagem proposta. Esta extensão inclui:
  - Testar a eficiência da replicação do índice.
  - Utilizar um conjunto maior de dados para indexar ou uma base da dados conhecida como o TREC[1].
  - Realizar testes com uma base dados real do Ustore e com seus usuários.
- **Expansão dos termos da consulta:** A abordagem pode ser evoluída para permitir a expansão dos termos da consulta. Esta expansão poderá, por exemplo, melhorar as taxas de precisão e *recall* obtidas.
- **Arquivos versionados:** O Ustore tem a capacidade de manter todas as versões dos arquivos modificados. Na abordagem proposta, n ao levamos isso em consideração. Então, é necessário estender a abordagem para que sejam mantidas todas as versões do conteúdo dos arquivos modificados.
- **Evitar *flooding* na rede:** Como discutido anteriormente, a busca na rede pode causar *flooding* se a quantidade de mensagens necessária para consultar todos os clientes Ustore for muito grande. Há na literatura diversas propostas para evitar este tipo de problema [Li and Wu, 2006], como: *include iterative deepening* [Yang and Garcia-Molina, 2002], *k-walker random walk* [Lv et al., 2002], *two-level k-walker random walk* [Jawhar and Wu, 2004], *intelligent search* [Kalogeraki et al., 2002].
- **Definir critérios para escolha dos clientes Servidores de Busca para serem consultados:** Todas as buscas foram realizadas tendo como base a escolha aleatória de clientes e Servidores de Busca. Alguns critérios podem ser adotados para definir quais seriam os melhores clientes e Servidores para responder a consulta, como: quantidade de informação indexada, tempo de resposta, histórico de respostas relevantes, etc.

[1] http://trec.nist.gov/ Acessado em: 11/08/2013

- **Recomendação de arquivos baseado no conteúdo:** Através dos metadados definidos é possível montar uma proposta para recomendação de informação baseada, por exemplo, no conteúdo dos arquivos, a categoria dos mesmo, as áreas de interesse do usuário.
- **Segurança dos arquivos indexados:** A abordagem proposta não abrange atributos relacionados com a segurança das informações armazenadas e como realizar consultas seguras neste ambiente. Há propostas relacionadas com esta área que podem ser adotadas como: [Wang et al., 2010; Cao et al., 2011; Wang et al., 2012].

Referências

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., AND ZAHARIA, M. 2009. Above the Clouds : A Berkeley View of Cloud Computing Cloud Computing : An Old Idea Whose Time Has (Finally) Come. *Computing* pp. 7–13.

ASSAD, R. E., MACHADO, M. A. S., SOARES, P. F. A., SILVA, A. F., SILVA, T. J. E., GARCIA, V. C., TRINTA, F., AND MEIRA, S. 2012. Desafios em cloud computing: Armazenamento, banco de dados e big data, pp. 76–111. *In* Tópicos em Multimídia, Hipermídia e Web. Sociedade Brasileira de Computação.

BAEZA-YATES, R. A. AND RIBEIRO-NETO, B. 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

BAROLLI, L. AND XHAFA, F. 2011. Jxta-overlay: A p2p platform for distributed, collaborative, and ubiquitous computing.

BERRIOS, D. C. 2000. Automated indexing for full text information retrieval. *In* Proceedings of the AMIA Symposium, p. 71. American Medical Informatics Association.

CAO, N., WANG, C., LI, M., REN, K., AND LOU, W. 2011. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *In* INFOCOM, 2011 Proceedings IEEE, pp. 829–837. IEEE.

DATTA, A., WU, D., XIN, L., AND WIERZBICKI, A. 2010. Handbook of Research on P2P and Grid Systems for Service-Oriented Computing. IGI Global.

DENG, J., HU, J., LIU, A. C. M., AND WU, J. 2010. Research and Application of Cloud Storage. *2010 2nd International Workshop on Intelligent Systems and Applications* pp. 1–5.

DUARTE, M. 2010. Um algoritmo de disponibilidade em sistemas de backup distribuído seguro usando a plataforma peer-to-peer. Master's thesis, Centro de Informática/UFPE.

DURÃO, F., ALMEIDA, E., AND LEMOS MEIRA, S. 2008. Applying a semantic layer in a source code retrieval tool. Master's thesis, Centro de Informática/UFPE.

DURÃO, F., ASSAD, R., FONSECA, A., FERNANDO, J., GARCIA, V., AND TRINTA, F. 2013. Usto.re: A private cloud storage software system. *In* F. Daniel, P. Dolog, and Q. Li (eds.), Web Engineering, volume 7977 of *Lecture Notes in Computer Science*, pp. 452–466. Springer Berlin Heidelberg.

GANTZ, J. AND REINSEL, D. 2011. Extracting Value from Chaos State of the Universe : An Executive Summary. pp. 1–12.

GARCIA, V., LUCRÉDIO, D., DURÃO, F., SANTOS, E., ALMEIDA, E., MATTOS FORTES, R., AND LEMOS MEIRA, S. 2006. From specification to experimentation: A software component search engine architecture, pp. 82–97. *In* I. Gorton, G. Heineman, I. Crnković, H. Schmidt, J. Stafford, C. Szyperski, and K. Wallnau (eds.), Component-Based Software Engineering, volume 4063 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

GONG, L. AND OTHERS 2001. Project JXTA: A technology overview. Technical report, Technical report, SUN Microsystems, April 2001. http://www. jxta. org/project/www/-docs/TechOverview. pdf.

HATCHER, E. AND GOSPODNETIC, O. 2004. Lucene in Action (In Action series). Manning Publications Co., Greenwich, CT, USA.

HEISS, J. J. 2005. Jxta technology brings the internet back to its origin. Technical report, Oracle.

HIEMSTRA, D. 2009. Information Retrieval Models. John Wiley and Sons, Ltd.

HUPFELD, F., CORTES, T., FOCHT, E., HESS, M., MALO, J., MARTI, J., CESARIO, E., KOLBECK, B., AND STENDER, J. 2008. The XtreemFS architecture – a case for object-based file systems. *Concurrency and computation: Practice and experience* 20:2049–2060.

HUPFELD, F., CORTES, T., KOLBECK, B., STENDER, J., FOCHT, E., HESS, M., MALO, J., MARTI, J., AND CESARIO, E. 2007. XtreemFS: a case for object-based storage in Grid data management. *In* 3rd VLDB Workshop on Data Management in Grids, co-located with VLDB, volume 2007, pp. 1–10.

JAIN, R. 1991. The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling. Wiley.

JAWHAR, I. AND WU, J. 2004. A two-level random walk search protocol for peer-to-peer networks. *In* Proc. of the 8th World Multi-Conference on Systemics, Cybernetics and Informatics, pp. 1–5.

KALOGERAKI, V., GUNOPULOS, D., AND ZEINALIPOUR-YAZTI, D. 2002. A local search mechanism for peer-to-peer networks. *Proceedings of the eleventh international conference on Information and knowledge management - CIKM '02* p. 300.

KAPLAN, J., ROY, R., AND SRINIVASARAGHAVAN, R. 2008. Meeting the demand for data storage. Technical report, The McKinsey Quarterly.

KOSTOFF, R. N. 2010. Expanded information retrieval using full-text searching. *Journal of Information Science* 36:104–113.

LENK, A., KLEMS, M., NIMIS, J., TAI, S., SANDHOLM, T., AND ALTO, P. 2009. What ' s Inside the Cloud ? An Architectural Map of the Cloud Landscape. *Management* pp. 23–31.

LI, X. AND WU, J. 2006. Searching techniques in peer-to-peer networks. *...Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks* pp. 1–31.

LOEST, S. R., MADRUGA, M. C., MAZIERO, C. A., AND LUNG, L. C. 2009. Backupit: An intrusion-tolerant cooperative backup system. *Computer and Information Science, ACIS International Conference on* 0:724–729.

LU, J. 2007. Full-text federated search in peer-to-peer networks. PhD thesis, School of Computer Science, Carnegie Mellon University.

LU, J. AND CALLAN, J. 2003. Content-based retrieval in hybrid peer-to-peer networks. *Proceedings of the twelfth international conference on Information and knowledge management - CIKM '03* p. 199.

LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. 2002. Search and replication in unstructured peer-to-peer networks. *In* Proceedings of the 16th international conference on Supercomputing, pp. 84–95. ACM.

MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.

MATTMANN, C. AND ZITTING, J. 2011. Tika in Action. Manning Publications Co., Greenwich, CT, USA.

MELL, P. AND GRANCE, T. 2009. The NIST Definition of Cloud Computing. Technical report.

OLIVEIRA, M. 2007. Ourbackup: Uma solução p2p de backup baseada em redes sociais. Master's thesis, Universidade Federal de Campina Grande.

PALANKAR, M. R., IAMNITCHI, A., RIPEANU, M., AND GARFINKEL, S. 2008. Amazon s3 for science grids: a viable solution? *In* Proceedings of the 2008 international workshop on Data-aware distributed computing, DADC '08, pp. 55–64, New York, NY, USA. ACM.

PEDDEMORS, R. AND SPOOR, R. 2010. Cloud Storage and Peer-to-Peer Storage: End-user considerations and product overview.

PETERSON, W. W. AND WELDON, E. J. 1972. Error-correcting codes. The MIT Press.

POUWELSE, J., GARBACKI, P., EPEMA, D., AND SIPS, H. 2005. The bittorrent p2p file-sharing system: Measurements and analysis, pp. 205–216. *In* Peer-to-Peer Systems IV. Springer.

RABIN, M. O. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM* 36:335–348.

RIJSBERGEN, C. J. V. 1979. Information Retrieval. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

ROBERTSON, S. AND ZARAGOZA, H. 2009. The probabilistic relevance framework: BM25 and beyond. Now Publishers Inc.

ROBERTSON, S. E. 1977. The probability ranking principle in ir. *Journal of documentation* 33:294–304.

SILVA, A., MACHADO, M., SOARES, P., JAMIR, T., GARCIA, V., AND ASSAD, R. 2012a. Desenvolvendo aplicativos peer-to-peer (p2p) no contexto de data storage para ambientes de cloud computing. *In* S. B. de Computação (ed.), Simpósio Brasileiro de Sistema de Informação.

SILVA, A., MACHADO, M., SOARES, P., JAMIR, T., GARCIA, V., ASSAD, R., AND MEIRA, S. 2012b. Usto.re uma plataforma descentralizada para storage de conteúdo em nuvens privadas. *In* CBSoft2012 - Industria.

SPARCK JONES, K., WALKER, S., AND ROBERTSON, S. E. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 1. *Information Processing & Management* 36:779–808.

STEINACKER, A., GHAVAM, A., AND STEINMETZ, R. 2001. Metadata standards for web-based resources. *IEEE Multimedia* 8:70–76.

STENDER, J., KOLBECK, B., HÖGQVIST, M., AND HUPFELD, F. 2010. BabuDB: Fast and Efficient File System Metadata Storage. *2010 International Workshop on Storage Network Architecture and Parallel I/Os* pp. 51–58.

TANG, C. AND DWARKADAS, S. 2004. Hybrid global-local indexing for effcient peer-to-peer information retrieval. *In* Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1, NSDI'04, pp. 16–16, Berkeley, CA, USA. USENIX Association.

VAQUERO, L. M., RODERO-MERINO, L., CACERES, J., AND LINDNER, M. 2009. A Break in the Clouds : Towards a Cloud Definition. *Computer Communication Review* 39:50–55.

VOGELS, W. 2008. A head in the clouds - the power of infrastructure as a service. *First workshop on Cloud Computing and in Applications (CCA '08)* .

WANG, C., CAO, N., LI, J., REN, K., AND LOU, W. 2010. Secure ranked keyword search over encrypted cloud data. *In* Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, pp. 253–262. IEEE.

WANG, C., CAO, N., REN, K., AND LOU, W. 2012. Enabling secure and efficient ranked keyword search over outsourced cloud data. *Parallel and Distributed Systems, IEEE Transactions on* 23:1467–1479.

YANG, B. AND GARCIA-MOLINA, H. 2002. Improving search in peer-to-peer networks. *In* Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on, pp. 5–14. IEEE.

YANG, Y., DUNLAP, R., REXROAD, M., AND COOPER, B. F. 2006. Performance of Full Text Search in Structured and Unstructured Peer-to-Peer Systems. *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* pp. 1–12.

ZENG, W., ZHAO, Y., AND OU, K. 2009. Research on cloud storage architecture and key technologies. *Technology, Culture and Human* pp. 4–8.