2.2. Computação em Nuvem

Na seção 2.2.1 são descritas as características essenciais, os modelos de serviço e os tipos de nuvem. Na seção 2.2.2 são abordados desafios relacionados à área de banco de dados quando aplicada à computação em nuvem. Por fim, na seção 2.2.3 são detalhados paradigmas de armazenamento de dados na nuvem.

2.2.1. Características, Modelos de Serviço e Tipos de Nuvem

Segundo o NIST (*National Institute of Standards and Technology* (http://www.nist.gov/itl/cloud/), a computação em nuvem possui cinco características essenciais, três modelos de serviços e quatro tipos de nuvem. Quanto às características essenciais, elas são:

- Serviço sob demanda: acesso direto e sob demanda, garantindo que a alocação e a liberação de recursos ocorra sem a necessidade de interação humana entre o usuário e o provedor.
- Acesso via rede: amplo acesso aos recursos oferecidos pela nuvem, o que garante uma alta disponibilidade desses recursos.
- Compartilhamento de recursos: oferecimento de recursos computacionais compartilhados entre diversos usuários, os quais não precisam ter conhecimento acerca da localização dos recursos que estão utilizando. Esses recursos computacionais devem ser abstraídos por dispositivos físicos reais, o que é alcançado na maioria das vezes por meio de virtualização.
- Elasticidade: rápida alocação e liberação de recursos da nuvem a qualquer momento, conforme a demanda da aplicação, de forma que o usuário não se preocupe sobre a quantidade de recurso a que tem direito e tenha a sensação de capacidade de armazenamento infinita, podendo a qualquer momento requisitar recursos de pico.
- Serviço mensurável: oferecimento de recursos mensurados, de forma que seja possível saber exatamente quais recursos da nuvem foram utilizados e que o usuário pague apenas por esses recursos utilizados. O conceito de serviço mensurável contribuiu para o surgimento do modelo *pay-as-you-go*.

Os recursos da nuvem podem ser oferecidos de acordo com diferentes modelos, os quais têm como principal diferença os tipos de recursos oferecidos. Na infraestrutura como serviço (IaaS), o tipo de recurso oferecido consiste em processamento, o qual é disponibilizado por meio de máquinas virtuais que são acessadas de forma transparente. Na plataforma como um serviço (PaaS), o recurso oferecido é uma plataforma para a execução de aplicativos, de forma que os aplicativos sejam desenvolvidos usando uma linguagem de programação que tenha suporte no provedor e sejam colocados para executar nos recursos na nuvem. Em contrapartida, no software como serviço (SaaS), que é o modelo de mais alto nível entre os modelos de serviço,

o recurso oferecido consiste em aplicações que executam em uma infraestrutura de computação em nuvem. Além desses três serviços definidos pelo NIST, é comum encontrar tanto na academia quanto na indústria um quarto modelo que define o armazenamento de dados como serviço (DaaS). Nesse modelo de serviço, o usuário tem acesso a um servidor de dados no qual ele pode requisitar espaço de armazenamento sob demanda [Sousa et al. 2010]. Por fim, do ponto de vista de ambientes de DWing, está surgindo o termo inteligência do negócio como um serviço (BIaaS) [Schneider et al. 2011].

Com relação à implantação da nuvem, ela pode ser feita em diferentes tipos de ambientes, variando desde ambientes totalmente privados a ambientes totalmente públicos [Armbrust et al. 2010, Dillon et al. 2010]. Basicamente, os tipos de nuvem se diferenciam pelas restrições de acesso: público, comunitário ou privado. Existe ainda a nuvem híbrida, na qual dois tipos de nuvem podem coexistir.

Do ponto de vista de implementação, um modelo de abstração que tem sido amplamente usado para a computação em nuvem é o *map-reduce* [Dean and Ghemawat 2004]. Nesse modelo, deve ser especificada uma função *map* que processa um par chave/valor para gerar um conjunto de pares chave/valor intermediários, e uma função *reduce* que junta todos os valores intermediários associados à mesma chave intermediária. Esse modelo explora as características da computação em nuvem, desde que foi concebido vislumbrando paralelismo, execução em *clusters* de máquinas e escalabilidade. Ademais, ele esconde detalhes de distribuição, paralelismo, tolerância a falhas e balanceamento de carga da implementação, além de possibilitar o gerenciamento de grandes volumes de dados.

2.2.2. Armazenamento de Dados em Nuvens

Dentro do contexto de banco de dados, um aspecto relevante da computação em nuvem refere-se ao armazenamento de dados em nuvens. Em [Brewer 2000] foi introduzida a ideia, provada posteriormente em [Gilbert and Lynch 2002], de que não é possível construir sistemas computacionais descentralizados que garantam ao mesmo tempo as propriedades de: (i) consistência: todas as réplicas dos dados possuem o mesmo valor a qualquer momento; (ii) disponibilidade: o sistema continua a operar se pelo menos um nó estiver em funcionamento; e (iii) tolerância a partições: a perda de mensagens entre os nós não afeta o funcionamento do sistema.

A impossibilidade de garantir essas três propriedades ao mesmo tempo é conhecida como teorema CAP (*Consistency*, *Availability* e *Partition Tolerance*). Entretanto, juntamente com o teorema CAP, Gilbert and Lynch (2002) provam que é possível garantir quaisquer duas dessas propriedades ao mesmo tempo. Portanto, o projeto de sistemas para execução em ambientes de computação em nuvem deve escolher duas propriedades em detrimento da terceira, decisão essa que deve ser tomada de acordo com os requisitos do sistema. Nesse sentido, uma tendência observada nos últimos anos pelos desenvolvedores de aplicações em nuvem é a escolha por disponibilidade e tolerância a partições, em detrimento da consistência.

Entretanto, surge um problema com os sistemas de gerenciamento de banco de dados tradicionais pois, em geral, esses foram desenvolvidos com o objetivo de garantirem as propriedades ACID, as quais envolvem a garantia da consistência dos dados. Para contornar esse problema, novas definições de consistência têm sido propostas e testadas, sendo que a mais difundida é a *consistência eventual*. Nesse sentido, as propriedades que têm se sobressaído na literatura são denominadas BASE (*Basically Available*, *Soft state*, *Eventually constistent*) [Pritchett 2008].

Um dos pontos fundamentais do BASE é, portanto, garantir consistência eventual, o que significa que uma requisição de leitura logo após uma requisição de escrita pode ter como retorno o valor antigo. Considere uma operação op_1 de escrita no tempo t_1 que ocorreu no nó n_1 e que alterou o dado do valor x para y, e uma operação op_2 de leitura no tempo t_2 que foi atendida pelo nó n_2. Com base nas propriedades BASE, é possível que a operação op_2 tenha como resposta o valor x (ou seja, valor antigo) ao invés do valor y (ou seja, novo valor), dependendo do tempo decorrido entre t_1 e t_2. Entretanto, é garantido que se o tempo entre op_1 e op_2 for suficiente, op_2 terá como resposta o valor y.

2.2.3. Paradigmas de Armazenamento de Dados na Nuvem

Uma vez que a consistência dos dados é usualmente flexibilizada para desenvolvimento de aplicações em nuvem, os sistemas gerenciadores de banco de dados tradicionais que utilizam o conceito de dados normalizados providos pelo modelo relacional deixam de ser interessantes. Com isso, novos paradigmas de armazenamento de dados começaram a surgir na literatura, visando atender à demanda das aplicações em nuvem. Os principais paradigmas, bem como exemplos de sistemas de gerenciamento de dados que os utilizam são [Sousa et al. 2010]: chave-valor, documento, coluna e grafo, além da proposta de adaptação do modelo relacional na nuvem Microsoft SQL Azure Database.

O paradigma chave-valor é o mais simples. O armazenamento dos dados ocorre em uma estrutura de dados conhecida como DHT (*Distributed Hash Table*), na qual os dados são representados apenas por conjuntos de chaves e valores associados. Dentre os paradigmas descritos nessa seção, o chave-valor é o que em geral proporciona o melhor desempenho para aplicações na nuvem. Entretanto, é também o de menor capacidade de busca, uma vez que permite apenas busca por chave. Em especial, objetos são usualmente indexados por chaves, naturais ou artificiais, cujo valor é o conteúdo serializado de um objeto. Portanto, não existe o suporte a consultas por atributos específicos. Exemplos de sistemas de gerenciamento de dados do tipo chave-valor incluem Redis (http://www.redis.io), Riak (http://www.riak.com), LevelDB (http://www.code.google.com/p/leveldb) e Voldemort (http://www.project-voldemort.com/).

O paradigma baseado em documentos oferece um modelo de dados livre de esquemas, ou seja, permite que diferentes instâncias de uma entidade possuam diferentes estruturas de

dados. A maneira em que os dados são armazenados, como o nome sugere, é por meio de documentos, em geral no formato XML ou no formato JSON, ou ainda em versões otimizadas desses formatos. Esse paradigma tem como vantagem permitir a realização de consultas um pouco mais elaboradas do que o paradigma chave-valor e ainda eficientes, uma vez que em geral é possível se construir índices sobre atributos mais importantes. Como destaques de sistemas de gerenciamento de dados dessa categoria, pode-se citar o MongoDB (http://www.mongodb.org) e o CouchDB (http://www.couchdb.apache.org).

O paradigma baseado em colunas consiste no modelo mais complexo e também mais semelhante ao modelo relacional, ao menos em seu conceito. Uma coluna é definida como um conjunto chave, valor e *timestamp*. Também são definidos os conceitos de *super coluna* como uma coluna especial, que contém outras colunas (mas não outras super colunas), e de uma *família de colunas*, que determina como os dados são armazenados em disco. As colunas da mesma família de colunas são armazenadas no mesmo conjunto de arquivos, ideia semelhante ao de uma tabela no modelo relacional. Entretanto, diferentemente de tabelas, não há um esquema fixo, de forma a possibilitar que quaisquer conjuntos de valores possam ser armazenados. Os principais sistemas de gerenciamento de dados que empregam o paradigma de colunas são o HBase (http://hbase.apache.org), o BigTable [Chang et al. 2006] e o Cassandra (http://cassandra.apache.org), esse último sendo uma mistura de paradigmas de armazenamento em chave-valor e coluna.

Como o próprio nome sugere, o paradigma baseado em grafos utiliza uma estrutura de dados de grafos para a representação dos dados. Nesse paradigma, os relacionamentos entre os elementos são tão importantes quanto seus dados em si. Como resultado, aplicações naturais do uso do paradigma baseado em grafos incluem a representação de amizades entre pessoas de uma rede social, um motor de recomendações de produtos relacionados em uma loja virtual, linked data (dados no formato RDF da Web Semântica) e a representação da *web*. Os principais sistemas de gerenciamento de dados baseados em grafos são o neo4j (http://www.neo4j.org) e o graphDB [Güting 1994].

Por fim, o Microsoft SQL Azure Database é um serviço na nuvem da plataforma Windows Azure (http://www.microsoft.com/brasil/windowsazure/sqlazure). Trata-se de um banco de dados relacional baseado na nuvem que estende as capacidades do Microsoft SQL Server. Ele pode armazenar dados estruturados, semi-estruturados ou sem nenhuma estrutura, sendo um dos seus principais recursos sua capacidade de persistência de dados relacionais. Outras características do Microsoft SQL Azure Database é que ele oferece um serviço de banco de dados altamente disponível, escalonável e tolerante a falhas.