**Problema:**

Estaciona.me é uma aplicação que informa a disponibilidade de vagas (mapeamento) em determinados estacionamentos através de um cadastro do usuário que pode ser feito via formulário no aplicativo/site ou Facebook. A mesma carece por outras funcionalidades extras e, à vista disso, identifica-se o uso de um banco de dados NoSQL para garantir um ambiente atrativo, acessível e confortável de modo que se torne essencial para o cliente.

Com a assistência do banco de dados NoSQL, o Estaciona.me auxiliará na escolha da vaga, indicando a mesma conforme dados coletados no facebook. Por exemplo, João e Maria são namorados e aos domingos, depois do almoço, gostam de ir ao cinema. Maria têm o costume de publicar seu status no Facebook identificando o filme que escolheram ver. Através destas informações, o Estaciona.me auxiliará João e Maria indicando a vaga disponível próxima ao cinema, isso em razão dos dados coletados do Facebook anteriormente.

Outra funcionalidade é a de filtrar o mapa conforme o motivo da visita do cliente. Será disponibilizado na página da aplicação um filtro que indicará as vagas próximas à: cinema, mercado, farmácia, praça de alimentação, boliche, caixas eletrônicos, etc. Estas informações serão armazenadas para uma análise de sentimento futuro e se tornarão indicações para escolha de vagas nas próximas visitas ao estabelecimento.

**Modelo de Dados NoSQL:**

Ideal para a aplicação, o Estaciona.me dispõe do banco de dados Columnar.

O banco de dados baseado em coluna permite particionar os dados, cria coleções de um ou mais pares de chave(s)/valor(es) que correspondem a um registro, não necessita uma estrutura de tabela criada anteriormente, são escaláveis, possuem consultas rápidas, indexação, compressão e garantem diferentes estratégias para executar o Business Intelligence e consultas para análise de sentimento do Estaciona.me.

**Cassandra:**

O Estaciona.me utilizará o Cassandra. Algum dos motivos adotados:

· Alta escalabilidade Linear: É possível adicionar mais recursos computacionais para aumentar o processamento. Além disso, se dobrarmos o número de nodos, dobramos sua performance.

· Armazenamento Descentralizado: Cada nodo tem uma parte dos dados. A hierarquia está ligada ao keyspace, família de colunas e a coluna.

· Sistema de Tolerância a Falhas: Sua replicação dos dados através das réplicas e que são armazenadas em múltiplos nodos. Também possui hash consistente onde cada dado é identificado através de uma chave e que serão distribuídos por uma função de hash.

· Alta taxa de disponibilidade: É possível enviar vários registros em uma mesma família de colunas ao mesmo tempo.

· Sem relacionamento: Uma boa prática para usar o Cassandra é desnormalizando sua base, desse modo, não existe relacionamentos. O mais correto seria uma família de colunas contendo as duas informações.

· Buscar informações pela chave: Dentro do Cassandra, por padrão, o único campo no qual se pode buscar informações é a chave. Para adicionar mais campos, basta defini-los como índice, mas isso não é bom por questão de performance.

· Cassandra Query Language: Para facilitar o desenvolvimento, é usado o Cassandra Query Language (CQL), com o qual pode-se criar e modificar estruturas e realizar manipulações de dados de uma maneira mais tranquila e muito mais fácil. O interessante é que esse recurso possui uma sintaxe muito semelhante ao SQL. Para executar e verificar os CQLs usamos o DataStax DevCenter.

· Coleções: O banco pode fazer uso de três coleções: o list (uma lista de informações), o set (uma lista sem valor duplicado) e um map (um dicionário de dados que possui um valor para uma chave correspondente). Não deve ultrapassar dos 260KB.

· Acompanhe seus nós: É possível acompanhar a performance e a topologia do seu data center através do DataStax OpsCenter. Em sua configuração, o recomendado é que o commitlog e o sstable estejam em discos diferentes e que esses sejam SSDs. É importante ter cuidado com o tamanho do heap, pois na maioria dos casos o padrão resolverá (é feito um cálculo de heap baseado em memória disponível).

**Arquitetura:**

**Memória:**

Quanto mais memória tiver um nó de Cassandra, melhor será o desempenho de leitura. A quantidade de memória RAM também permite tabelas de memória (memtables) para armazenar mais recentemente dados escritos. Os memtables maiores levam a um número menor de SSTables sendo liberados para o disco, mais dados mantidos no cache da página do sistema operacional e menos arquivos para serem varridos do disco durante uma leitura.

Utiliza-se em um primeiro momento, a memória recomendada para o Estaciona.me:

Produção: 128 GB;

Desenvolvimento em ambientes de teste sem carga: 32 GB.

**CPU:**

As cargas de trabalho com inserção pesada são vinculadas à CPU em Cassandra antes de se tornarem vinculadas à memória. Altamente concorrente e usa tantos núcleos de CPU como disponíveis, todas as gravações vão para o log de confirmação, no entanto o Cassandra é tão eficiente na escrita que a CPU é o fator limitante.

Utiliza-se, em um primeiro momento, CPU recomendada para o Estaciona.me:

Hardware dedicado em ambientes de produção: processadores de CPU de 16 núcleos para manter o equilíbrio atual de preço-desempenho;

Hardware dedicado no desenvolvimento em ambientes de teste sem carga: processadores de CPU de 2 núcleos.

**SSD:**

Utiliza-se, em um primeiro momento, SSD recomendada para o Estaciona.me:

1TB SATA III 6Gb/s

Leituras: 540MB/s e Gravações: 520MB/s.

**Rede:**

Como Cassandra é um armazenamento de dados distribuído, ele coloca carga na rede para lidar com solicitações de leitura/ gravação e replicação de dados entre nós.

Utiliza-se a largura de banda de 1000 Mbs.

**Espaço em Disco:**

Utiliza-se 2 discos por node, um para o log de commit e outro para os diretórios. Além disso, é recomendado 2TB por nó de espaço em disco para dados não compactados.