Arquitetura em Big Data para organização de milhões de ligações em uma empresa de Telecom

Osilmar Mendonça Costa Filho[1], Marcelo de Castro Cardoso [2]

Curso de Engenharia de Computação
Centro Universitário de Anápolis (UniEvangélica) – Anápolis, GO – Brazil

osilmar@outlook.com, marcelo.cardoso@docente.unievangelica.edu.br

Abstract. *With the exponential growth of data volume, there is a need to provide high availability of this data. For this availability are proposed architectures directed to Big Data, with the use of non-relational database (NoSQL). Big Data provides many tools and technologies to facilitate development. Thus, the proposal of this work is to present an architectural solution to solve the problem of a Telecom company, in the storage and generation of reports on millions of connections.*

Resumo. *Com o crescimento exponencial do volume de dados, há a necessidade de fornecer alta disponibilidade destes dados. Para esta disponibilidade são propostas arquiteturas voltadas para Big Data, com a utilização de banco de dados não relacionais (NoSQL). Big Data fornece muitas ferramentas e tecnologias para facilitar o desenvolvimento. Assim, a proposta deste trabalho é apresentar uma solução arquitetural para solucionar o problema de uma empresa de Telecom, no armazenamento e geração de relatórios sobre milhões de ligações.*

1. Introdução

Nas últimas décadas a preocupação com o crescimento de dados não era tão evidente, o armazenamento e recuperação ou processamento destes dados em sistemas de banco de dados relacional era suficiente. Mas ainda hoje encontramos situações similares, porém nos últimos anos este cenário vem se modificando, os dados crescem exponencialmente, segundo Taurion [2012] chegamos a casa de *petabytes* gerados diariamente. E processar e analisar todos estes dados está cada vez mais difícil.

Um dos grandes problemas em lidar com grandes volumes de dados em sistemas de bancos de dados é a exigência de recursos computacionais [Vieira 2012]. E para resolver este problema há duas soluções: crescimento na vertical, onde se adquire máquinas maiores e com mais potência em processadores, memória e armazenamento em disco - o que torna o custo deste cenário muito elevado; ou crescimento na horizontal – com a utilização de *clusters* de máquinas menores utilizando *hardwares* mais accessíveis,

[1] Graduando em Engenharia de Computação no Centro Universitário de Anápolis – UniEvangélica.

[2] Mestre em Ciências da Computação - UFG e Professor no Curso Engenharia da Computação da UniEvangélica.

com um custo mais baixo. Possibilitando a distribuição das tarefas através deste *cluster*, ganhando mais poder computacional.

Big Data é o armazenamento extenso de dados, que tem por base o conceito 5V's; No início dos anos 2000 Doug Laney um famoso analista do setor de Big Data, articulou e definiu os 3V's sendo eles [SAS], o volume de dados - é realmente grande e o crescimento é de forma exponencial; a velocidade - Segundo Taurion [2012a] a importância da velocidade é porque muitas vezes é preciso agir praticamente em tempo real, sobre grande volume de dados ; a variedade - os dados podem ser gerados de forma estruturada e não estruturada de diversos lugares: rede sociais, *e-mail*, vídeos, sensores e outros; a veracidade - verificar veracidade e consistência dos dados; e por fim valor - "Quanto maior a riqueza de dados, mais importante é saber realizar as perguntas certas no início de todo processo de análise." [BROWN, ERIC, 2015 apud NETTO].

Arquiteturas em *Big Data* podem fornecer soluções capazes de analisar grande volume de dados em tempo real. Para esta análise utiliza-se banco de dados não relacionais também conhecidos como *NoSQL*.

As tecnologias da *Big Data* são acessíveis para pequenas e medias empresas, trazendo muitos benefícios. Os resultados obtidos através destas tecnologias, nos apresentada dados que podem melhor a maneira de relacionamento com os clientes. [Pensando Grande, 2016].

Utilizando como estudo de caso uma empresa do segmento de *Telecom*, a qual solicitou que seu nome não fosse divulgado, tal empresa atualmente enfrenta problemas relacionado ao crescimento constante de seus dados. O que acabou gerando um "gargalo" no desempenho do sistema.

Ao analisar os requisitos levantados: escalabilidade, modelagem do banco e baixo desempenho verificou-se que o sistema passa por um "gargalo" muito grande na geração de relatórios e fechamento de faturas, devido ao grande volume de dados armazenados e sua arquitetura.

A tabela de ligações tem o crescimento diário de 80.000 a 90.000 registros, totalizando por volta de 2 400.000 registros ao mês. No fechamento da fatura, e consequente geração de relatório, a tabela ligações relaciona-se com diversas outras – gerando assim um alto custo computacional. Mesmo com a reestruturação do MER (Modelo Entidade Relacionamento) o custo computacional ainda será alto devido ao volume de dados e seu crescimento constante, tornando assim a análise dos dados em tempo real inviável.

Os dados desta empresa atualmente encontram-se em uma base de dados relacional, e o mesmo apresenta problemas de estrutura e desempenho. Uma possível solução seria o crescimento na horizontal, mas segundo Fowler [2013] banco de dados relacionais não foram projetados para serem utilizados em *clusters*.

Com base neste problema será proposto um modelo de arquitetura *Big Data* com a utilização de modelo de banco de dados *NoSQL* capaz de armazenar grande volume de dados que possibilite solucionar o gargalo, que forneça suporte a consulta e agregação com menor tempo possível, em relação a variabilidade e o volume de dados estruturados.

2. Trabalhos Relacionados

Algumas literaturas mostram que *Big Data* pode trazer expressivas incursões nas empresas, proporcionando novas oportunidades. Existe três importantes informações a serem extraído dos dados: redução de custo, melhorias no processo decisórios e melhoria no produto e serviços.

Em 2003, Oren Etzione ao viajar de Seattle para Los Angeles para visitar seu irmão, comprou sua passagem meses antes acreditando que iria pagar mais barato no Bilhete, ao perguntar à pessoa ao lado por curiosidade o valor que tinha pagado e quando havia comprado, constatou que é homem havia pagado consideravelmente um valor menor do que ele, mesmo a passagem sendo comprada recentemente, ao consultar outros passageiros também foi constatado que a maioria deles havia pagado menos. A partir deste momento Etzione percebeu uma série de problema de dados, determinado a descobrir uma maneira das pessoas saberem se o preço das passagem on-line seria ou não um bom negócio. Em 41 dias ele consegui uma amostragem de dados de 12 mil preços obtidos através de sites de viagens. Seu pequeno projeto evoluiu para uma *startup* chamada *Farecast*, que após conseguir um dos banco de dados de reserva da indústria aeroviária, a *Farecast* agora utilizava uma base de dados de 200 bilhões de registro de preço para suas previsões, em 2008 a *Microsoft* adquiriu a *Farecast* por cerca de US$110 milhões e integrou a seu sistema de busca do *Bing*, o sistema acertava 75% das previsões trazendo assim uma economia em média de US$50 por passagem a seus usuário. [Cukier 2013]

Em 2011 a *Xoom* uma empresa especializada em transferência financeiras e associadas a empresas que fazem analises de *Big Data*, disparou um alarme ao notar que uma quantidade de transações da *Discover Card* estava ligeiramente maior que a média em *New Jersey*. As transações pareciam legítimas mais vinham de um grupo de criminosos, só foi possível a detecção uma vez que todos os dados haviam sido examinados. [Cukier 2013].

A operadora Vivo cada vez mais vem ampliando seus esforços em *Business Intelligence* e *Big Data*, no ano de 2015 a empresa aumentou em 18 vezes sua capacidade de processamento de informação, para o ano de 2016 esperasse que esta capacidade seja multiplicada para 87 vezes. A operadora está utilizando *Big Data* para melhora sua oferta de planos móveis (pré, pós e controle), ainda utilizando, técnica de estatística *Machine Learning* de computação cognitiva, para gerar *insghts* através de analises das informações podendo assim oferecer a seus clientes novos portfólio de serviços que se adeque ao seu perfil de uso. Outro projeto que já vem gerando resultados é a identificação e solução de problemas massivos na rede fixa (voz e dados), realizado no início de 2016, o projeto consegui reduzir em 70% o tempo de identificação e solução, atingindo 85% de precisão nos diagnósticos, reduzindo custo no *Call Center* como com as equipes de campos. [Telesintese 2016].

Outra operadora que vem intensificado seus avanços em *Big Data* é a Tim, a operadora quer aproveitar a expertise no tratamento de 6 bilhões de registros de usuários ao dia, com a análise e relatórios de comportamento de usuários, a empresa espera prever as tendências de seus clientes. O teste inicial utilizando a plataforma de *Big Data* da Tim realizado com terceiros e com parceria com a Prefeitura do Rio de Janeiro, aconteceu durante as Olimpíadas, foram contabilizados mais de 2,6 bilhões de dados da Tim e mais de 1,6 milhão de dados de turista em suas redes, sendo possível a Tim, prover à Prefeitura

informações importantes sobre mobilidade urbana, nacionalidade dos turistas e locais onde eles se deslocaram e maior concentração de pessoas. [Cordeiro 2016].

A *T-Mobile* obteve um alcance de 50% em sua taxa de cancelamento, após analisar, buscar e visualizar a base de dados de seus assinantes, os objetivos do negócio era ter uma visão em 360 graus para reconhecer em tempo hábil problemas relacionados com seus clientes, trazer uma redução de custos e ampliar sua cadeia de inovação, conhece por quais canais seus clientes preferem se comunicar. As análises comportamentais incluíram conhecer como seus clientes utilizam seus produtos, quantas vezes, onde e qual tempo médio dura as ligações de seus clientes entre outras. Com analise de chamadas foi possível identificar que ao mudar para uma área diferente o cliente pode receber uma cobertura limitada, então um alerta é feito para que um dos seus representantes, assim ele pode oferecer num novo serviço ou promoção, evitando que seu cliente migre para outra operadora. Nas análises de sentimentos incluíram indicadores de ações de seus clientes, era importante perceber o que eles pesavam sobre a *T-Mobile*. Através destas abordagens individualizadas e centrada em seus clientes, obteve-se uma queda significativa nos cancelamentos mensais, que antes, foram cerca de 100 mil no primeiro trimestre de 2011, para 50 mil clientes no segundo trimestre do mesmo ano. [Van Rijmenam].

Ao analisar todos trabalhos acima relatado e também outros casos de sucessos relacionados a *Big Data*, chega à conclusão que esta migração na arquitetura pode trazer muito benefícios, não somente no ganho computacional, mas também no conhecimento mais aprofundado dos clientes, as análises nestes dados podem trazer *insghts* que passa despercebido, esperasse que os valores apresentados possibilitem oferecer produtos e serviços mais objetivos e individual para o usuário e também um conhecimento maior sobre eles.

3. Big Data

As tecnologias para *Big Data* oferecem novas oportunidades em análise e compreensão mais aprofundada dos dados. Mapeando os padrões dos dados, sendo estruturados e não estruturados. Para este mapeamento utiliza-se ferramentas com grande poder computacional, e modelo de computação paralela.

Big Data mudou a forma de entender e explorar os dados, pois a análise dos dados é impulsionada a uma investigação não causal. *Big Data* tem o conceito de ampliar o potencial das analises, mas *Big Data* não é somente o volume de dados, está também ligada a variedade, ao acesso a outras fontes de dados, esta fonte geralmente é externa, possibilitando assim uma ampliação da visão do contexto.

Para Jill Dyché vice-presidente da *SAS Best Pratices*, *Big Data* é mais que apenas um grande volume de dados não estruturados, ele também conclui que as tecnologias que possibilita o processamento e análise, estas tecnologias possibilitam analisar conteúdo de forma rápida. [Davenport 2014].

O termo *Big Data* está cada vez mais popular, embora ainda esteja mal compreendido. Observa-se em muitas palestras que não existe consenso quanto a que realmente é *Big Data* e quais as tecnologias fundamentais que o sustentam. [TAURION 2012].

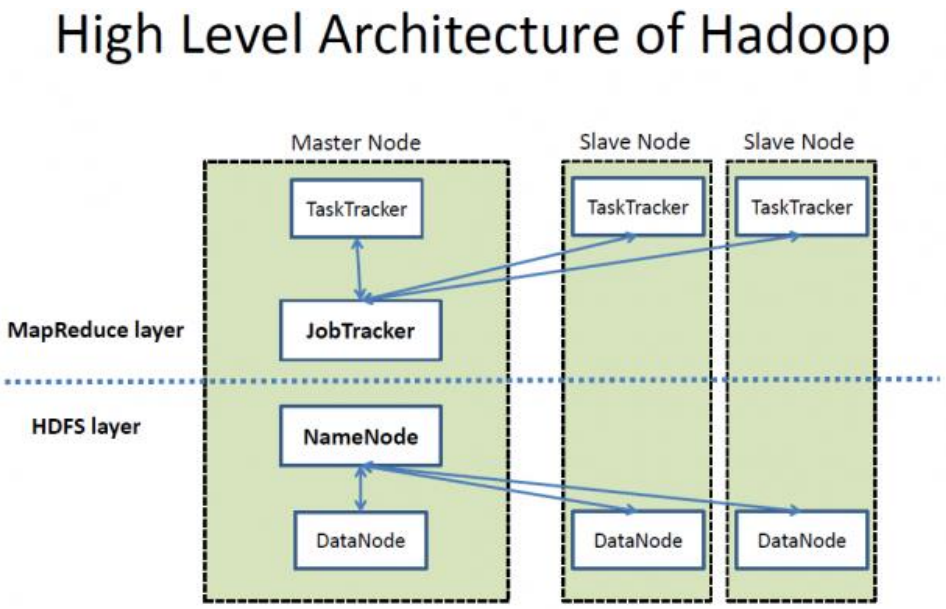
A seguir será abordado com mais detalhes ferramentas que auxiliam, no desenvolvimento de uma arquitetura voltada a *Big Data*, facilitando assim a compreensão dos dados.

3.1. Apache Hadoop

O *Hadoop* foi criado pelo *Yahoo* em 2005 e pode ser considerado uma das maiores invenções de data management desde o modelo relacional. [TAURION 2011].

O *Hadoop* é uma ferramenta para aplicação analítica em dados massivos e não estruturados. Na prática é a combinação de dois projetos *HMR* (*Hadoop MapReduce*) que é responsável pelo processamento distribuído e o *HDFS* que armazena grande volume de dados também de forma distribuída, conforme Figura 1.

Figura 1. Arquitetura do Hadoop.



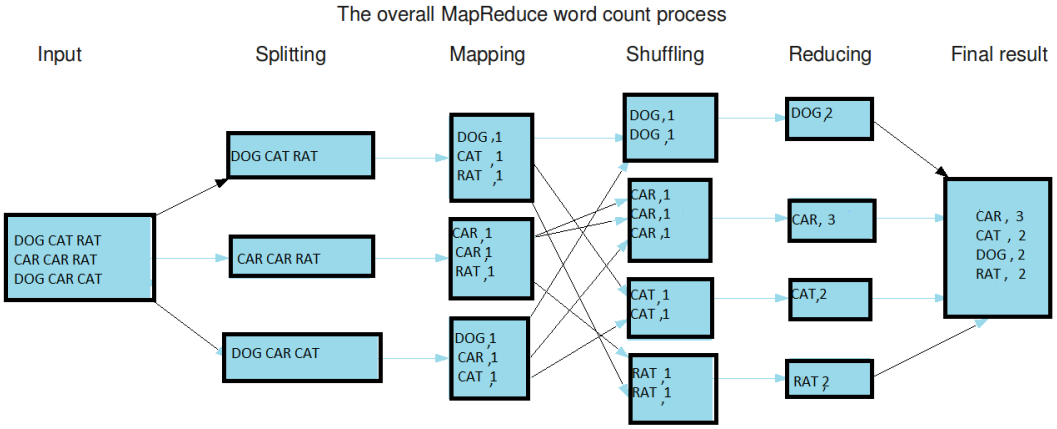
Fonte: https://opensource.com/life/14/8/intro-apache-hadoop-big-data

3.2. Hadoop MapReduce

Hadoop MapReduce é um modelo de programação paralela, muito utilizado para processamento de dados em larga escala. O processamento é dividido em pequenas tarefas independentes, através de um *cluster* de máquinas para melhorar o aproveitamento do processamento (Figura 2), após o processamento o retorno é único. O processamento é baseado em dois momentos, *Map* e *Reduce*.

O *Map* é o mapeamento dos dados de entrada que gera uma lista de pares <chave, valor>, os valores associados a uma chave são agrupados e enviado ao *Reduce*. O *Reduce* combina estes valores para formar um conjunto de dados e geraram um resultado final.

Figura 2. Exemplo de um processo MapReduce para contagem de palavras.



Fonte: http://a4academics.com/tutorials/83-hadoop/840-map-reduce-architecture

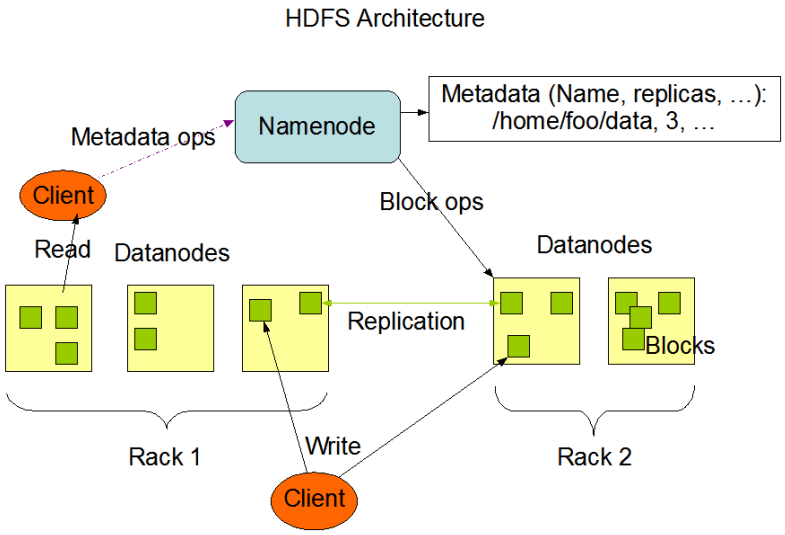
3.3. Apache HDFS

O *HDFS (Hadoop Distributed File System)* é um subprojeto *Apache Hadoop*, um sistema de arquivo altamente tolerante a falhas e projetado para executar em hardware de baixo custo. [Hanson 2013].

HDFS é um sistema de arquivo distribuído que tem sobre sua responsabilidade armazenar, localizar, compartilhar e proteger os arquivos que estão distribuídos entre os nós de um *cluster*, localizado na camada de armazenamento do *Hadoop.* Ele é responsável pelo alto desempenho na leitura e escrita de grandes arquivos, entre suas características estão a escalabilidade, disponibilidade e tolerância a falhas, uma vez que os dados são replicados em servidores diferentes. Isto se deve a arquitetura (Figura 3) estruturada em *master-slave* (mestre-escravo) conseguindo trabalhar com dois processos principais, são eles:

- ***Namenode*:** É responsável pelo gerenciamento dos dados (arquivo) que foram armazenados. Estas informações são organizadas através de metadados que contém os registros dos *Datanodes.* Os mesmos são responsáveis pelos blocos, mapeamento, divisão dos arquivos e o encaminhamento para os blocos de nós escravos.
- ***Datanode*:** Tem como sua responsabilidade o armazenamento dos conteúdos dos arquivos nos nós escravos. Cada nó pode armazenar vários blocos e fazer a replicação em diferentes máquinas, é comum a existência de várias instâncias de *Datanodes* em aplicações *Hadoop*.

Figura 3. Arquitetura HDFS.



Fonte: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

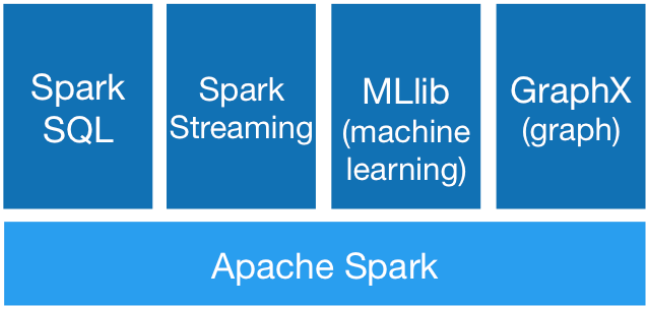
3.4. Apache Spark

O *Spark* é um *framework* para processamento de *Big Data* com foco na análise sofisticada, facilidade de uso e principalmente velocidade. Em clusters *Hadoop* as aplicações executam até 100 vezes mais rápidas em memória e até 10 vezes em disco. Suporta operações de *MapReduce*, consultas em *SQL*, *streaming* de dados, aprendizagem de máquina e processamento de grafos. [PENCHIKALA, 2015].

Os resultados de processamentos são retidos na memória, o que fornece velocidade maior, e a possibilidade de processar o mesmo dado várias vezes. O *Spark* armazenará a quantidade máxima de dados em memória antes de persisti-la em disco.

O Spark contém componentes específicos para cada tipo de processamento, embutidos no *Spark Core*, *map*, *reduce*, *filter* e *collect são* componentes e funções básica para o processamento. [SANTANA, 2016]. Na Figura 4 é apresentado o ecossistema *Spark*.

Figura 4. Ecossistema do Spark.



Fonte: http://spark.apache.org

Além de uma API, o ecossistema disponibiliza bibliotecas adicionas que fornecem capacidade adicional nas análises de Big Data e aprendizagem de máquina. Estas bibliotecas incluem:

- ***Spark Streaming***: Permite o processamento dos dados em streaming em tempo real.
- ***Spark SQL***: Permite a utilização de consulta no estilo *SQL* sobre os dados processado através de uma *API JDBC*.
- ***Spark MLlib***: Biblioteca de aprendizado de máquina do *Spark*, algoritmo de aprendizagem como regressão, *clustering*, filtragem colaborativa e redução de dimensionalidade.
- ***Spark Graphx***: Uma *API* para grafos de alto nível e computação paralela. Possui uma crescente coleção de algoritmos para simplificar as análises de grafos.

3.5. NoSQL

No final da década de 1990 foi utilizado o termo *NoSQL* pela primeira vez, por Carlo Strozzi, para um sistema de *SGBD* de código aberto desenvolvido por ele, que não utilizava *SQL* como linguagem de consulta. Somente em 2009, no evento produzido por Johan Oskarsson e Eric Evans, que tinha por objetivo discutir o crescimento de projetos *open source* de armazenamento de dados distribuído não relacional, o termo foi utilizado para descrever esta nova vertente de soluções de armazenamento. [FOWLER, 2013].

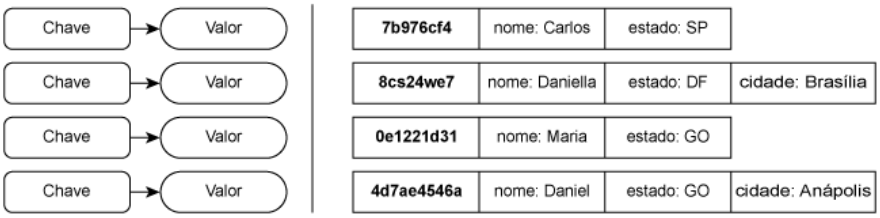
Umas das características principal deste modelo de banco dados é não ser relacional, mais nada impende o uso de linguagem *SQL*. Outra característica do *NoSQL* é não precisar de uma estrutura pré-definida para o armazenamento dos dados, isto é feito dinamicamente à medida que os dados vão sendo inseridos. Atualmente existe 4 tipos de categorias sendo: chave-valor, família de colunas, orientado a documentos e baseado em grafos.

3.5.1. Chave-Valor

Banco de dados de chave-valor é uma tabela *hash* simples, tendo somente duas colunas, a primeira sua chave primária e a segunda os valores a serem armazenados. Este modelo de banco de dados *NoSQL* é o modelo mais simples de ser utilizado por perspectiva de uma *API*.

Os dados podem ser obtidos, inseridos ou apagados através de uma determinada chave. Sendo de responsabilidade da aplicação entender o que foi armazenado. Na Figura 5 detalha o armazenamento chave-valor. “Banco de dados de chave-valor não se importam com o que é armazenado na parte do valor do par chave-valor. O valor pode ser um *blob*, texto, *JSON, XML* e assim por diante.”. [FOWLER, 2013].

Figura 5. Modelo de dados com chave-valor.



Fonte: COSTA FILHO, O. M.

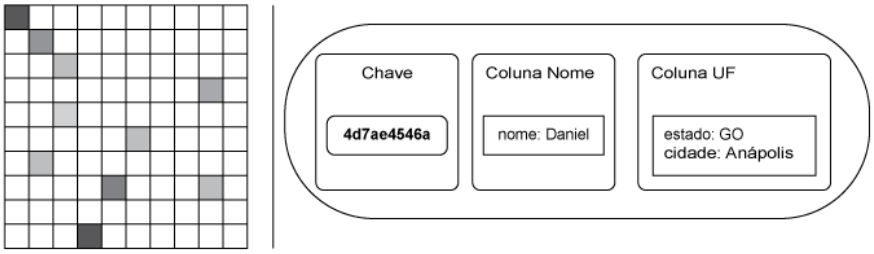
Os bancos mais populares são *Riak*, *Redis*, *MemcacheDB* e o *Amazon DynamoDB*, estes bancos são muito utilizados para cache pois mantém os conjuntos de dados em memória. [LEAVITT, 2010 apud FREIESS, 2013].

3.5.2. Família de colunas

O armazenamento em família de colunas é feito através de mapeamento de valores, onde estes valores são armazenados em famílias (Figura 6). Cada família suporta a agregação de bilhões de registro, sendo assim o modelo de banco de dados mais utilizado para *Data Warehouses*.

Em alguns casos este modelo de banco de dados não é muito recomendado. Um exemplo são sistemas que requerem muitas transações *ACID* para gravações e leituras. [FOWLER 2013].

Figura 6. Modelo de dados com famílias de colunas.



Fonte: COSTA FILHO, O. M.

Os bancos de dados mais populares são *HBase*, *Cassandra*, *Hypertable* e o *Amazon SimplesDB*.

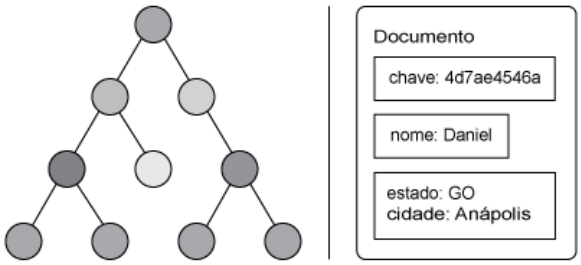
3.5.3.Orientado a Documentos

Este modelo de banco de dados tem como características o armazenamento, recuperação e a organização de modelos de dados semiestruturados, geralmente armazenados em formato *JSON*, *BSON* ou *XML*. (Figura 7)

Os documentos são armazenados através de uma chave única e o valor representa o documento. Estes documentos são semelhantes entre si, mais não necessitam serem exatamente iguais. “Os documentos permitem que novos atributos sejam criados sem a necessidade de definição prévia ou de alteração nos documentos existentes.”. [FOWLER 2013].

Por ter um esquema mais flexível não impõe restrições, uma vez que os dados são gravados como entidades da aplicação. Assim podemos realizar consultas de dados dentro do documento sem ter a necessidade de recuperar o documento inteiro por sua chave e depois examiná-lo.

Figura 7. Modelo de dados com documentos.



Fonte: COSTA FILHO, O. M.

Banco de dados mais populares são: *MongoDB*, *CouchDB* e o *RavenDB*.

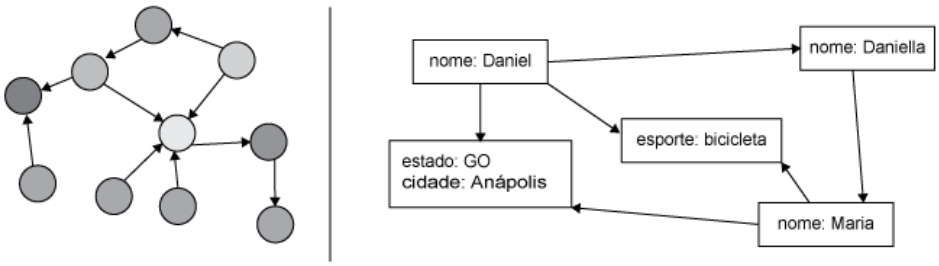
3.5.4.Baseado em Grafos

O banco de dados em grafos surgiu como uma alternativa ao banco de dados relacional para dar suporte a sistemas cuja interconectividade de dados é um aspecto importante. [PENTEADO].

Banco de dados de grafos, segue o conceito da teoria dos grafos, onde o grafo consiste em nós, propriedades e arestas. Os nós representam as entidades, as propriedades os atributos e as arestas os relacionamentos (Figura 8).

"As arestas têm significância direcional; nodos são organizados por relacionamentos, os quais permitem que você encontre padrões interessantes entre eles.". [FOWLER, 2013].

Figura 8. Modelo de dados em grafos.



Fonte: COSTA FILHO, O. M.

Banco de dados mais populares são: *Neo4J*, *Titan*, *OrientDB* e o *FlockDB*.

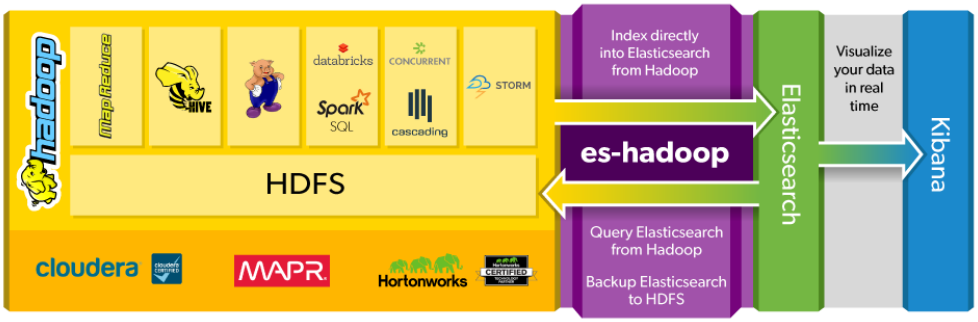
3.5.5.Elasticsearch

Elasticsearch é uma ferramenta distribuída para mineração e tratamento de dados, tem como principal função permitir que dados desestruturados possam ser armazenados de forma simples e eficiente. [HENRIQUE 2015].

O objetivo do *Elasticsearch* está no apoio ao desenvolvimento de aplicações centrada em texto. Sua arquitetura foi projetada para ser escalável com gerenciamento de grande volume dados de forma simples e eficiente. *Elasticsearch* tem como principal vantagem sua *engine* de pesquisa *full-text*, com alta disponibilidade permite buscar grandes volumes de dados em tempo real, indexados em forma de documentos e com uma capacidade analítica em computação distribuída. Seu motor de busca é baseado no *Apache Lucene*, que é utilizado para construir seu estado de pesquisa.

O *ES-Hadoop* é um conector de duas vias que permite a interação entre o *Elasticsearch* e o *Hadoop* (Figura 9), utilizando a capacidade analítica do *Elasticsearch* em conjunto com os dados armazenados no *Hadoop* para desenvolvimento de aplicações que trabalham com dados em tempo real, permitindo assim um melhor aproveitamento dos dois mundos.

Figura 9. Exemplo de utilização do Elasticsearch com Hadoop.



Fonte: https://www.elastic.co/products/hadoop

4. Proposta de Arquitetura

Existem diversas maneira de uma arquitetura em *Big Data* ser implementada, cada tecnologia é responsável por uma parte desta arquitetura, seja para armazenar, processar ou analisar os dados, a escolha de uma arquitetura é um desafio, pois é necessário considerar muitos fatores, um muito importante é avaliar se o cenário do negócio e um problema de Big *Data*. Aproveite os dados que não estão sendo usado para possíveis *insight.* Não ignore o potencial dos dados que tem em mãos.

Após ser avaliado o crescimento constante da tabela *calls* (Figura 10), a necessidade de melhorar o desempenho dos relatórios e também o fechamento de fatura dos clientes. Uma arquitetura em *Big Data* tem o poder computacional para melhor o desempenho dos requisitos acima, pode concluir que o cenário do cenário do negócio realmente é um problema de *Big Data*. Foi adotado uma implementação de forma

incremental, assim de neste primeiro momento seria resolver os problemas relacionado a tabela *calls*.

Figura 10. Crescimento de 2 362 700 tuplas.

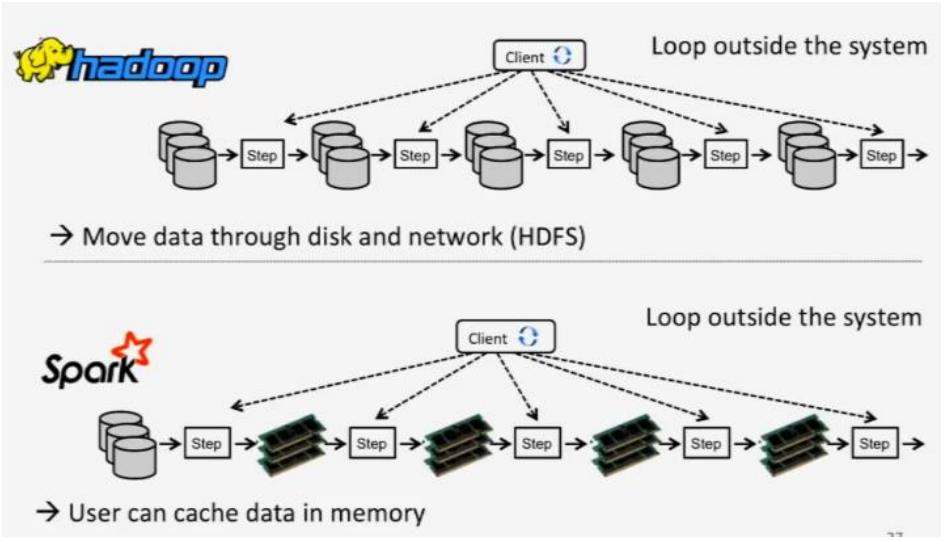
Name	Value
Name	calls
OID	19892
Schema	public
Database	backup.sgc2.2016.10.09
Owner	postgres
Group Name	
Rows	62329500
Table Type	Normal

Name	Value
Name	calls
OID	36277
Schema	public
Database	backup.sgc2.2016.11.10
Owner	postgres
Group Name	
Rows	64692200
Table Type	Normal

Fonte: COSTA FILHO, O. M.

Para o problema que espera-se resolver, foram definidas algumas tecnologias para *Big Data* sendo elas, aplicar o Hadoop e o Hadoop MapReduce para o tratamento de dados e o sistema de arquivos HDFS. Para o problema o Hadoop se mostrou mais robusto e performático. Mesmo o Spark se mostrando mais rápido em determinados cenários, ele consegue ser mais ágil que o Hadoop, pois trabalha com os dados em memória (Figura 11). Mas ao ser inserido um novo dado é necessário fazer uma reorganização destes dados, que em determinados caso pode-se elevar o custo de processamento. Aliada ao Hadoop será utilizado o Elasticsearch como engine de busca. O *Elasticsearch* armazena seus dados no modelo de documentos, tornando sua pesquisa de dados muito eficiente.

Figura 11. Hadoop & Spark.



Fonte: http://www.nextplatform.com/2015/02/22/flink-sparks-next-wave-of-distributed-data-processing/

A arquitetura proposta será implementada de forma incremental, e o *Hadoop* funciona em computação distribuída com *clusters*, e como a implementação / teste(s) inicial(is) está sendo realizada em uma máquina local (*desktop*) será configurado

posteriormente em um ambiente específico e favorável onde o tempo de execução de processamento dos dados terá efeito.

A utilização de um modelo de banco de dados *NoSQL* de documentos, mostrou-se mais indicado para análise de dados em tempo real. Partes dos documentos pode ser atualizadas facilmente, ainda podemos criar novas métricas. É um modelo de banco de dados que fornece diversos recursos de pesquisa. Que podem ser através de visões – pesquisa complexas em seus documentos que podem ser materializadas. Outro recurso muito interessante, é a pesquisa dos dados dentro do documento sem a necessidade de recuperar o documento todo para examiná-lo. A escalabilidade é feita através da adição de mais nodo escravos para leitura, pode-se adicionar mais capacidade ao cluster à medida que a carga vai aumentando. Na *Tabela 1* veremos um comparativo entre todos os modelos de banco de dados *NoSQL*.

***Tabela 1: Modelos de banco de dados* NoSQL**

Modelos *NoSQL*		
Chave-Valor	Uso apropriado	▪ Armazenamento de informações em sessão. ▪ Perfil de usuário, preferencias. ▪ Dados de carrinho de compras.
	Não utilizar	▪ Relacionamento entre os dados. ▪ Transeções com múltiplas operações. ▪ Consulta por dados. ▪ Operações em conjuntos.
Família de Colunas	Uso apropriado	▪ Registro de eventos (log). ▪ Sistema de gerenciamento de conteúdo (CMS). ▪ Contadores.

	Não utilizar	▪ Transações *ACID*. ▪ Agregação de dados em consulta (*SUM* ou *AVG*).
Orientado a Documento	Uso apropriado	▪ Registro de eventos (log). ▪ Sistema de gerenciamento de conteúdo (CMS). ▪ Análise web ou em tempo real (analytics). ▪ Aplicativos de comércio eletrônico.
	Não utilizar	▪ Transações complexas que abranjam diferentes operações. ▪ Consultas em estruturas agregadas variáveis.
Baseado em Grafos	Uso apropriado	▪ Dados Conectados. ▪ Roteamento, envio e serviço baseados em localização. ▪ Mecanismo de recomendação
	Não utilizar	▪ Atualizar todas entidades ou subconjuntos.

Fonte: Adaptada de Fowler 2013.

Para o ambiente de teste para comparativos de banco de dados relacional e *NoSQL*, foi feita através de uma máquina virtual com 2 núcleos de processamento e 10gb de memória *ram*, utilizou o *Red Hat* 7.3 x64 com sistema operacional além das tecnologias como *PostgreSQL* 9.5, *ElasticSearch* 5.0.0 com *Apache Lucene* 6.2.0, *Java* 1.8.0_111 e *dotNet* Core 1.0.0-preview2-003131.

Primeiramente foi realizado um estudo sobre a tabela *calls*, sua estrutura e seus relacionamentos (Figura 12), podendo assim ser mapeados os dados que realmente são de importância para os relatórios e a geração de faturas dos clientes.

Figura 12. Estudo da estrutura da tabela *calls*.

Fields | Indexes | Foreign Keys | Uniques | Checks | Excludes | Rules | Triggers | Options | Comment | SQL Preview

Name	Type	Length	Decimals	Not null	
id	int4	32	0	☑	1
number_id	int4	32	0	☑	
tariff_id	int4	32	0	☐	
tariff_operator_id	int4	32	0	☑	
bill_period_id	int4	32	0	☐	
line_consumption_id	int4	32	0	☐	
hawk_id	int4	32	0	☐	
start	timestamp	6	0	☑	
area_code	int4	32	0	☐	
state	varchar	2	0	☐	
called_number	varchar	16	0	☐	
call_period	varchar	255	0	☐	
duration_min	int4	32	0	☑	
duration_sec	int4	32	0	☑	
operator_value	numeric	12	2	☑	
created_at	timestamp	6	0	☐	
updated_at	timestamp	6	0	☐	
import_call_id	int4	32	0	☑	
cost_value	numeric	12	2	☐	
charged	bool	0	0	☑	
value	numeric	12	2	☐	
eagle_id	int4	32	0	☐	

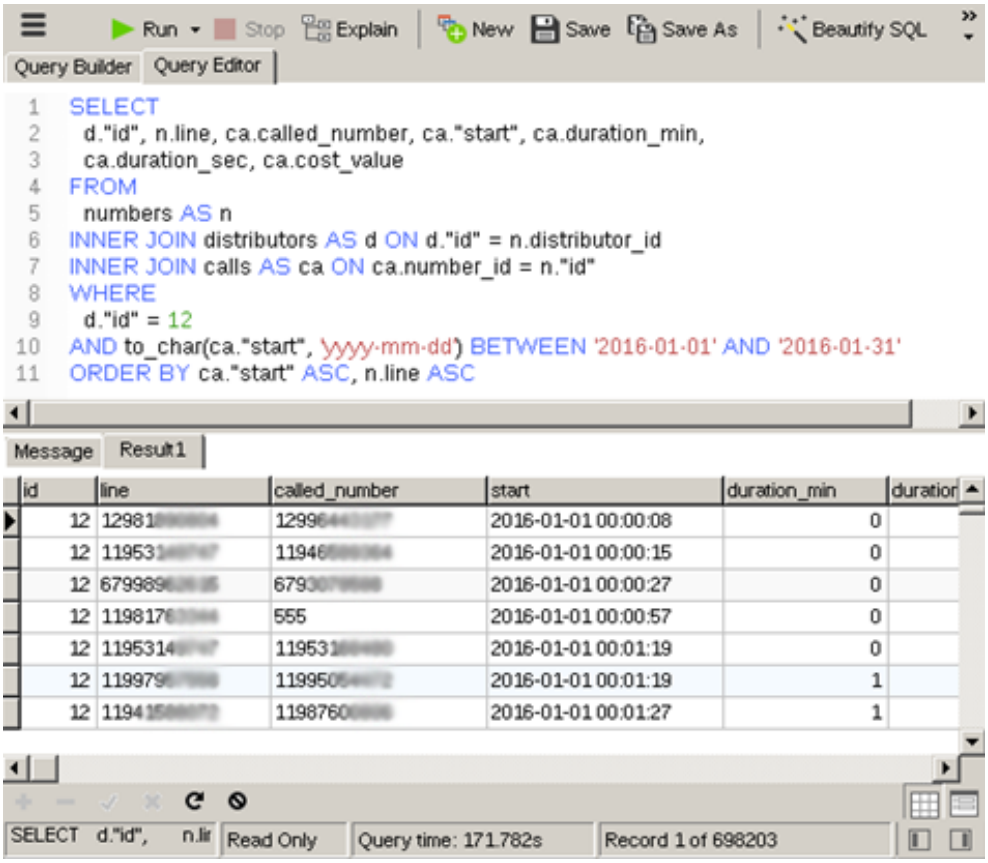
Fields | Indexes | Foreign Keys | Uniques | Checks | Excludes | Rules | Triggers | Options | Comment | SQL Preview

Name	Fields	Referenced Schema	Referenced Table	Referenced Fields	On Delete	On Update
calls_bill_period_id_fk	bill_period_id	public	bill_periods	id	NO ACTION	NO ACTION
calls_import_call_id_fk	import_call_id	public	import_calls	id	NO ACTION	NO ACTION
calls_line_consumption_id_fk	line_consumption_id	public	line_consumptions	id	NO ACTION	NO ACTION
calls_number_id_fk	number_id	public	numbers	id	NO ACTION	NO ACTION
calls_tariff_id_fk	tariff_id	public	tariffs	id	NO ACTION	NO ACTION
calls_tariff_operator_id_fk	tariff_operator_id	public	tariff_operators	id	NO ACTION	NO ACTION

Fonte: COSTA FILHO, O. M.

Avaliando o desempenho de um relatório simples, sobre a tabela *calls* com união a outras duas tabelas, como podemos verificar na figura (Figura 13), a consulta tem como objetivo lista as ligações realizadas apenas por um cliente entre um intervalo de datas, a pesquisa encontrou 698203 registros, levanto 171,782 segundos para ser finalizada. Sendo considerado somente o tempo de execução da consulta, o desempenho se mostrou insatisfatório. Uma vez que clientes que utilizam este tipo de relatórios vem constantemente reclamando do tempo de espera na geração dos relatórios, e como estes dados estão em constante crescimento, isto impacta cada vez mais na geração dos relatórios.

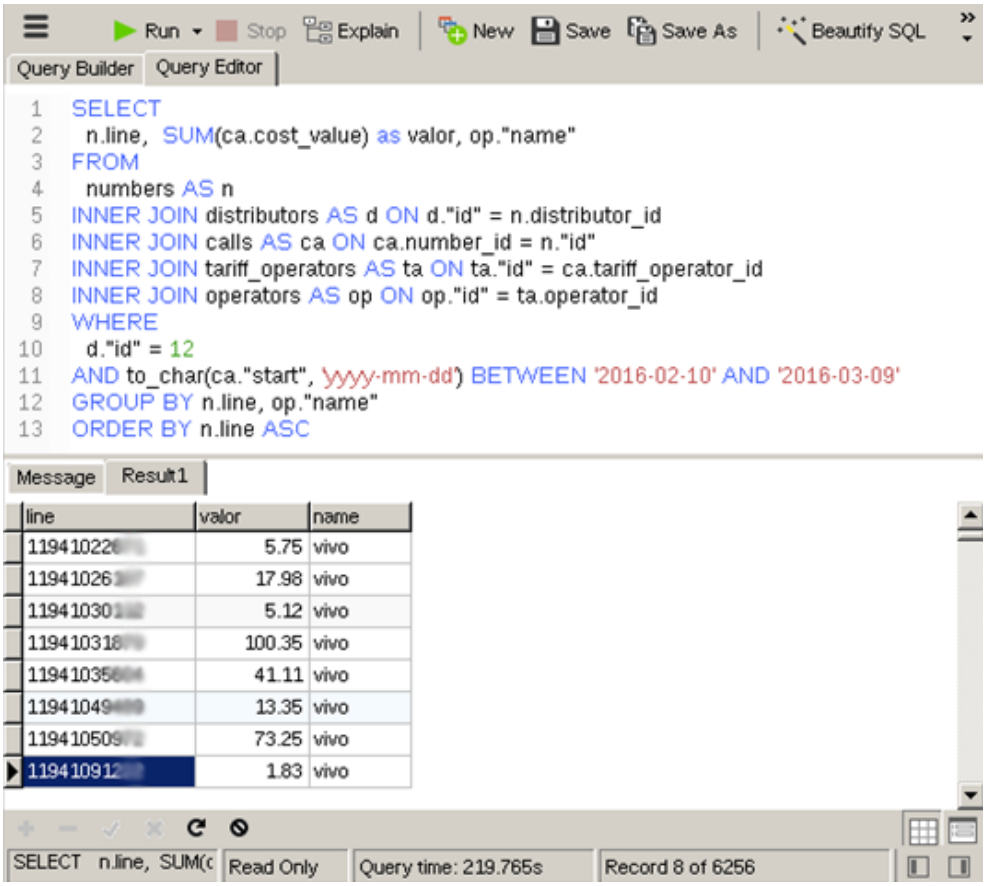
Figura 13. Consulta de ligações entre intervalos de data.



Fonte: COSTA FILHO, O. M.

Em outra consulta mais elaborada com a tabela *calls* em união com outras quatros tabelas, para informar o valor gasto por cada linha em um período de um determinado cliente, demorou 291,765 segundos (Figura 14) para obter os dados esperando, o tempo também foi bastantes insatisfatório de acordo com as premissas que foram citadas anteriormente. Pois os dados estão em uma máquina local, não havendo latência de rede, sendo assim ao ser colocando em ambiente de produção este tempos pode ainda mais expressivos.

Figura 14. Consulta de valor gasto das linhas, no intervalo de datas.



Fonte: COSTA FILHO, O. M.

A estrutura do modelo de dados em documento proposta tem como premissas otimizar o modelo do banco de dados relacional, assim selecionar os dados que realmente são importantes para os relatórios e fechamento de faturas. Sendo assim o documento contém outros quatros documentos internos, documento de com os dados do cliente - foram mantidos o *id* do cliente (*associate*) para não perder a referência junto aos dados que ainda continuará no banco de dados relacional - documento com os dados referente a origem da ligação, documento do destino da ligação e o documento de ligação - contém a data que foi realizado, duração da ligação e seus respectivos valores de acordo com as tarifas, como poderemos visualizar na Figura 15 a seguir.

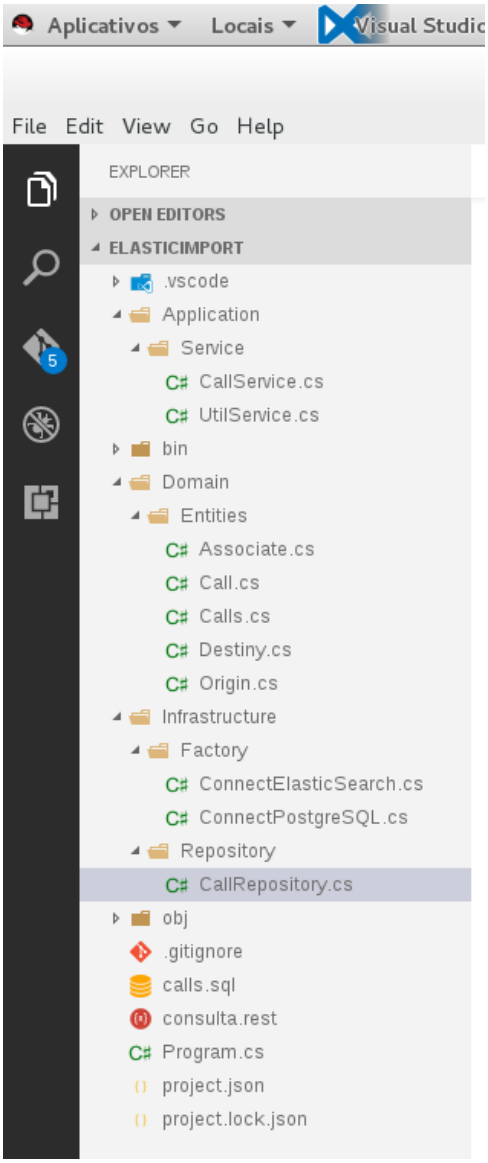
Figura 15. Estrutura proposta da tabela calls.

Fonte: COSTA FILHO, O. M.

A migração da tabela *calls* para o modelo não estruturado do *Elasticsearch*, ganha-se um poder computacional maior, sendo possível a utilização de *clusters* e trabalhar as consultas de formar distribuídas e paralelas.

Foi desenvolvido um sistema em *Batch* (Figura 16) utilizando a linguagem *C#*, com o objetivo de extrair dados do *PostgresSQL* (Banco de dados relacional), e migrar os dados para o *Elasticsearch*, ele recupera um bloco de linhas na tabela *calls* juntamente com as outras tabelas relacionadas, posteriormente converte os dados em um objeto *JSON*, no modelo de documento proposto acima, e por último persiste os dados os dados no *Elasticsearch*. Este sistema somente será utilizado na etapa inicial do projeto, depois que for concluída toda a migração destes dados, os outros sistemas que são responsáveis por inserir os dados na tabela *calls* será apontado para o próprio *Elasticsearch*.

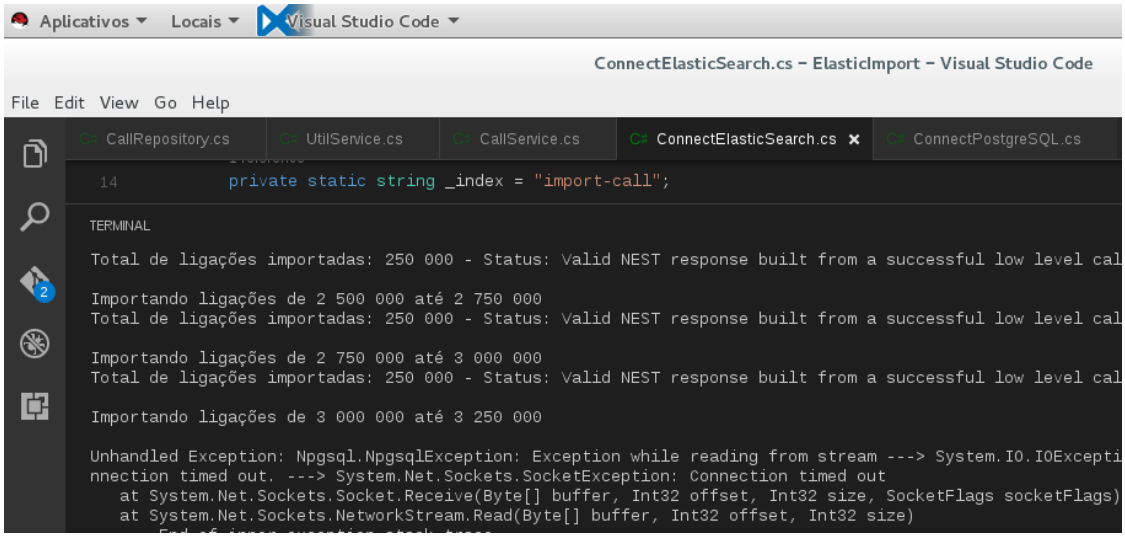
Figura 16. Arquitetura do sistema em C# para migração de dados.



Fonte: COSTA FILHO, O. M.

Ao longo do processo de migração dos dados, surgiu uma adversidade que relacionada a quantidade de registros importadas em cada bloco de dados. Os primeiros blocos abrangiam 250 000 registros, mas ao alcançar cerca de 3 000 000 de registro a consulta no banco de dados relacional obteve um desempenho crítico, o sistema foi automaticamente encerrado com a resposta de *timeout* como podemos visualizar na Figura 17.

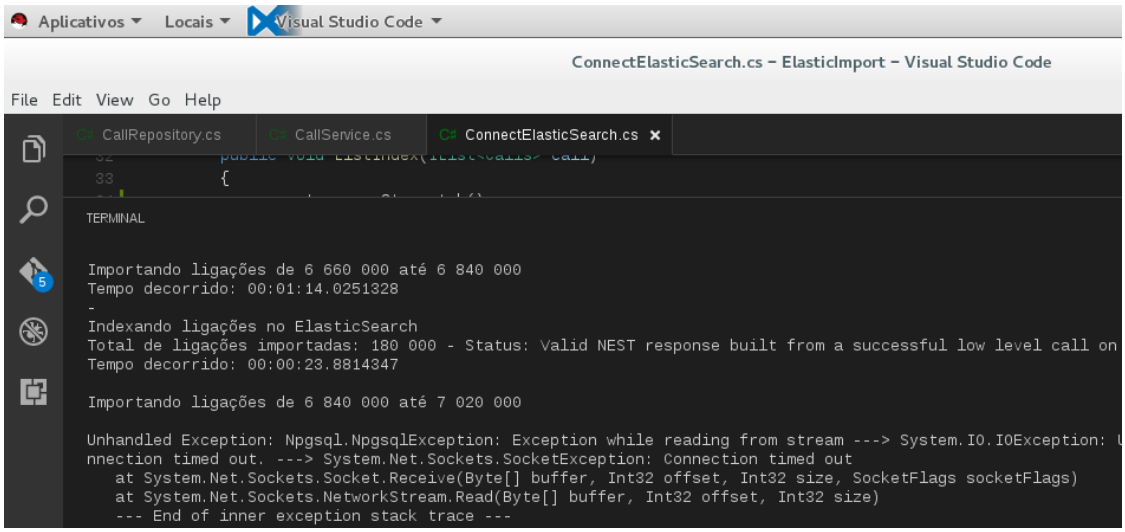
Figura 17. Adversidade na importação de dados com 250 000 registros.



Fonte: COSTA FILHO, O. M.

Em outra tentativa de migração dos dados do *PostgreSQL* para o *Elasticsearch*, foram diminuídos o tamanho do bloco de registro para 180 000 registros, a quantidade de dados importados aumentou, mas também ocorreu a mesma adversidade ao obter o total de 7 000 000 de registros importados (Figura 18).

Figura 18. Adversidade na importação de dados com 180 00 registros.

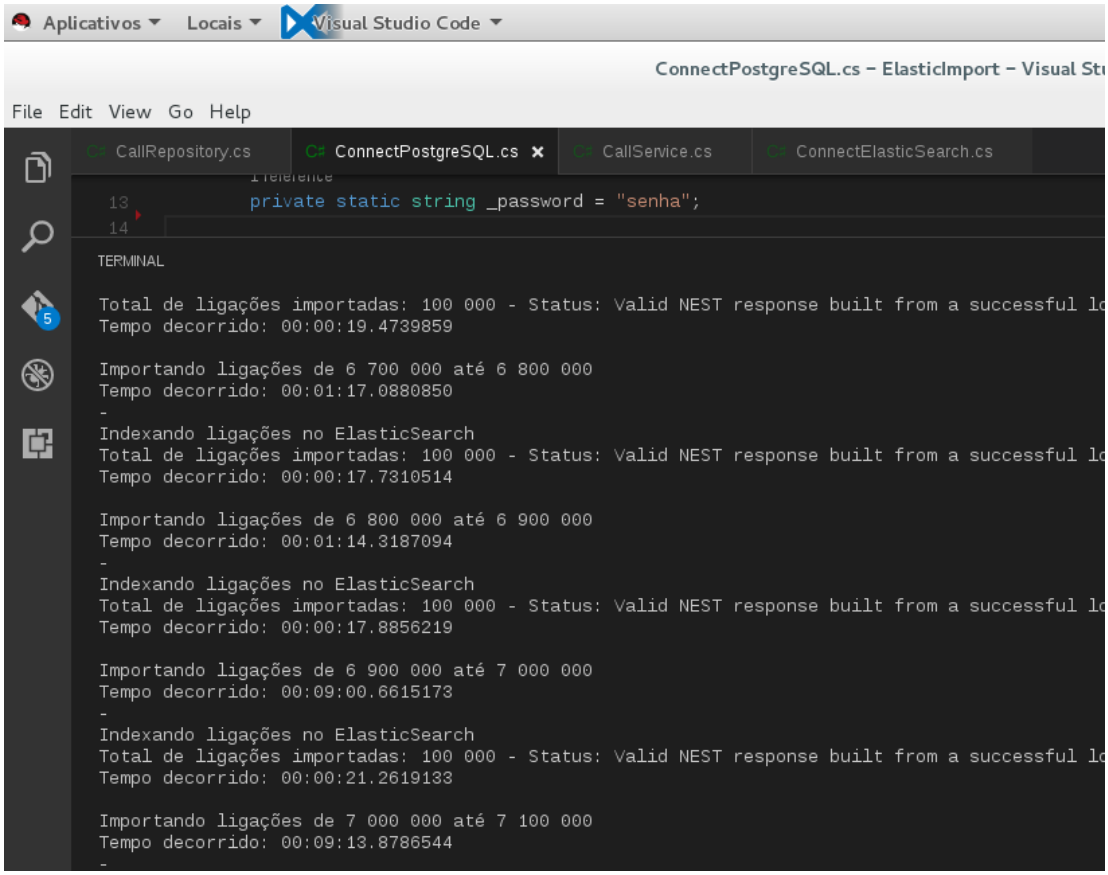


Fonte: COSTA FILHO, O. M.

A adversidade foi controlada ao se importar blocos de dados com 100 000 registos, porém ocorreu uma anomalia no tempo de importação, as consultas realizadas nos registros acima de 6 900 000, o tempo de pesquisa no *PostgreSQL* subiu de cerca de 1 minuto para cerca de 9 minutos (Figura 19). Essas anomalias de tempo eram de forma irregular pode ser maior ou menor que 9 minutos, mesmo as buscas no *PostgreSQL* se

mantendo sempre e bloco de 100 000 registros. Entretanto a importação dos dados foi concluída sem nenhuma ocorrência.

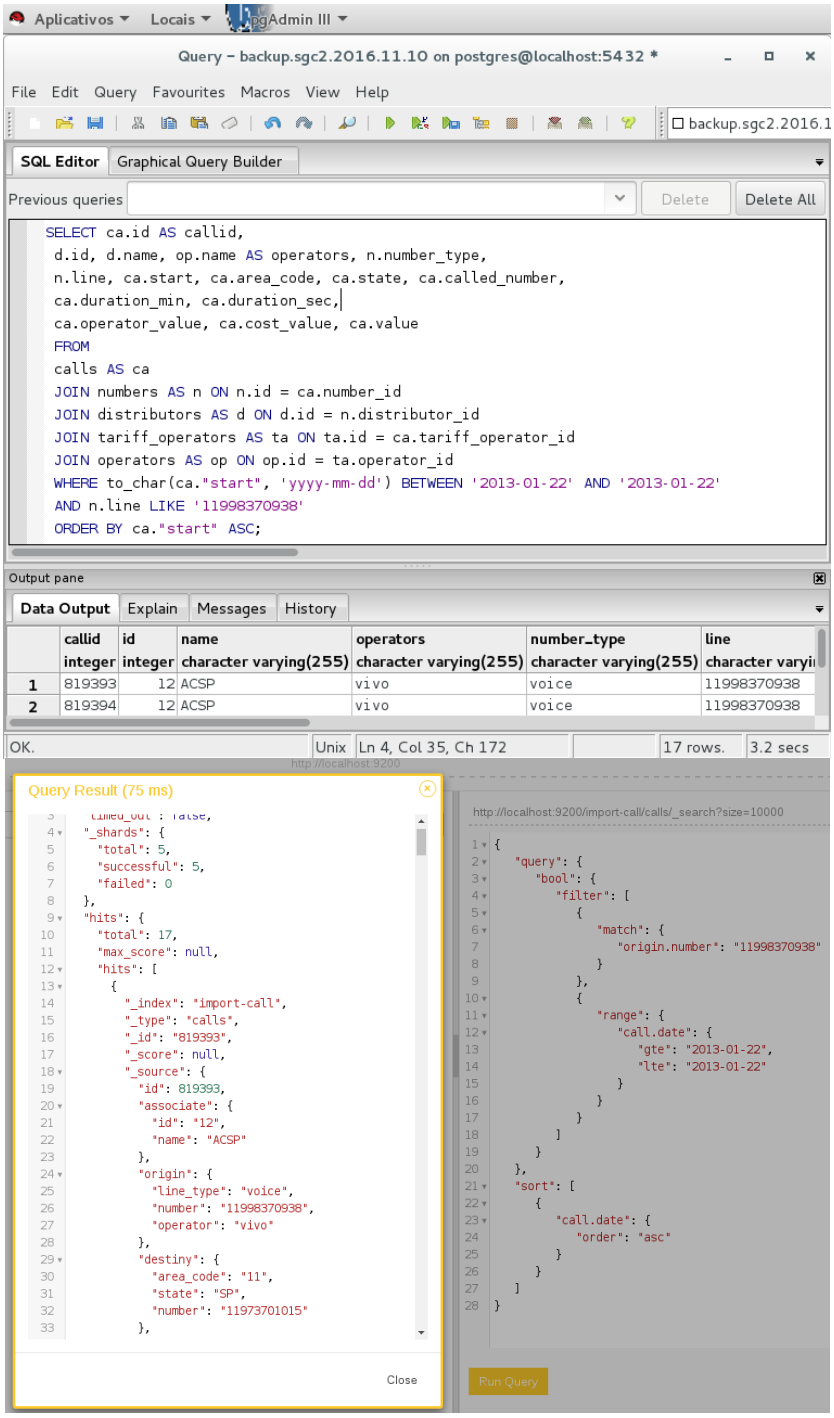
Figura 19. Anomalias na importação.



Fonte: COSTA FILHO, O. M.

Nas comparações realizada entre o banco de dados relacional e o *NoSQL*, foi constatado um notável ganho de desempenho do *Elasticsearch*, na (Figura 20) pode-se visualizar que mesmo em uma consulta bem simples, o *Elasticsearch* consegue obter resultados de forma mais performática. Na parte superior encontra-se a consulta realizada e no *PostgreSQL* com o resultado de 17 registro em um tempo de 3.2 segundos (3.200 milissegundos). Já na parte inferior da figura encontra-se a consulta realizada no *Elasticsearch* com o resultado de 17 registro em um tempo de 75 milissegundos, ambas consultas são idênticas tem como o mesmo objetivo de buscar ligações realizadas de uma linha em um dia.

Figura 20. Comparação de consulta no banco de dados relacional e *NoSQL*.

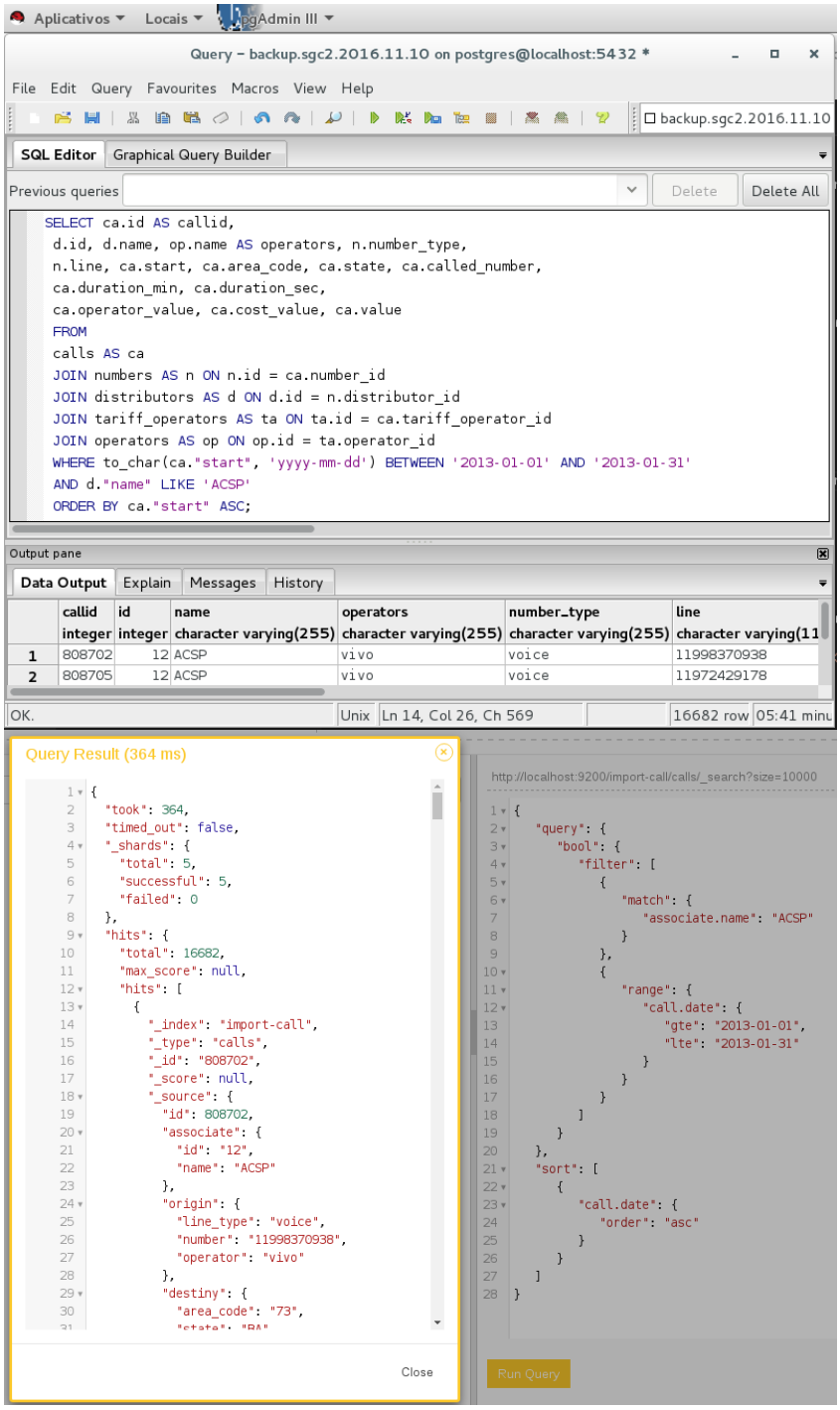


Fonte: COSTA FILHO, O. M.

Em outra consulta idêntica entre o *PostgreSQL* e o *Elasticsearch*, tento ambas os e mesmo objetivo de buscar todas as ligações de um cliente entre o intervalo de um mês, novamente o *Elasticsearch* se mostrou mais performático (Figura 21). Na parte superior encontra-se a consulta realizada e no *PostgreSQL* com o resultado de 16 682 registro em um tempo de 05:41 minutos (341 000 milissegundos). Já na parte inferior da figura

encontra-se a consulta realizada no *Elasticsearch* com o resultado de 16 682 registro em um tempo de 364 milissegundos.

Figura 21. Outra comparação de consulta no banco de dados relacional e NoSQL.



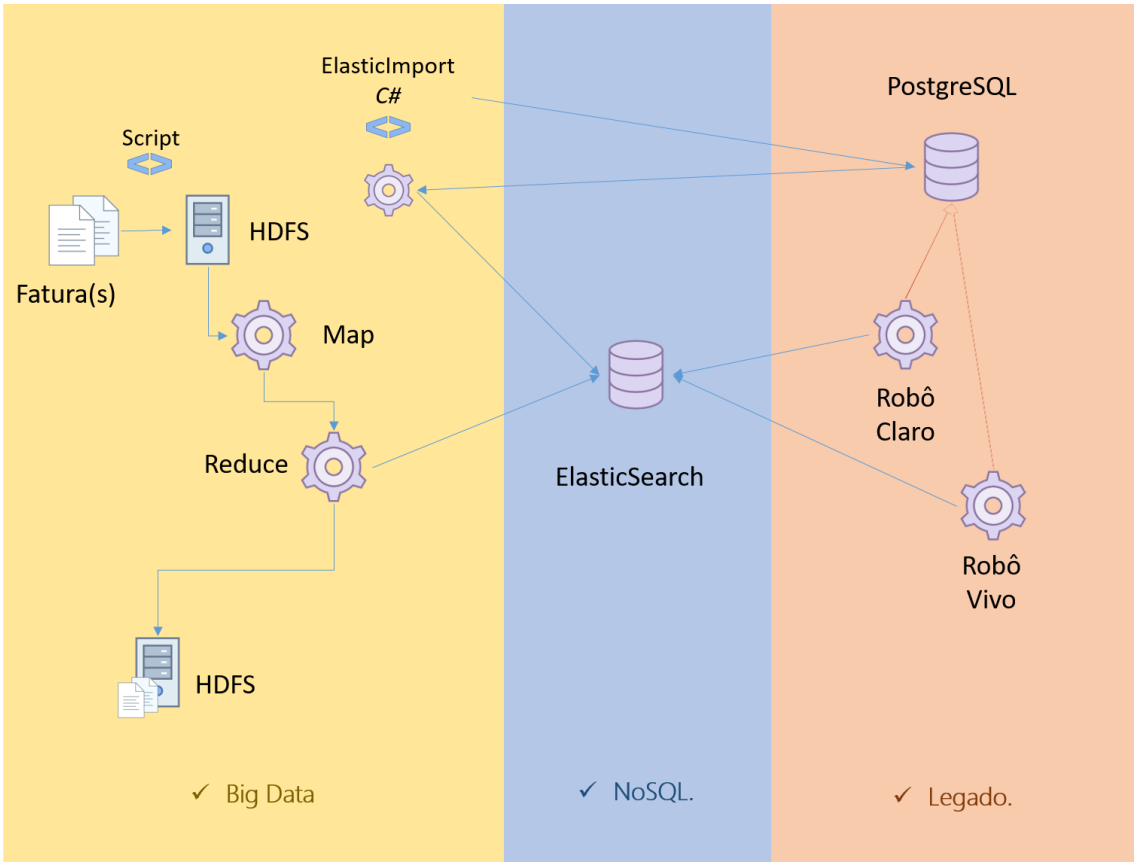
Fonte: COSTA FILHO, O. M.

A partir dos resultados obtidos nos comparativos entre os banco de dados relacional e *NoSQL*, podemos concluir que o modelo de banco de dados *NoSQL*, consegue atender o problema que foi ele foi proposto a resolver, melhorias no desempenho da

geração de fatura e geração de faturas dos clientes, vale a pena ressaltar que o *Elasticsearch* está configurado para rodar em *Standalone*, ou seja não está sendo executado de forma distribuída onde o mesmo tem um ganho do seu poder computacional, assim podemos concluir que em um ambiente propicio as mesma consultas podem tem um resultado ainda melhor, das que foram demonstrada.

A arquitetura proposta (Figura 22) se divide em três características principal que são elas: Primeira característica: *Big Data*, onde são feitas a leituras dos arquivos ligações cedidas pelas operadoras, seguido do *MapReduce* das informações contidas nos arquivos e posteriormente ao *Reduce* estes dados são inseridos no *Elasticsearch.* Segunda característica a utilização de banco de dados *NoSQL,* o modelo utilizado foi orientado a documentos. E a terceira e última característica os sistemas legados, que temos como exemplo o *PostgreSQL*, agentes que captura informações de ligações no site das operadoras.

Figura 22. Modelo arquitetural.



Fonte: COSTA FILHO, O. M.

Os sistemas legados não têm como ser descontinuado neste primeiro momento, uma vez que o *PostgreSQL* atualmente contém mais de 150 tabelas que mantem os dados de todos os clientes. Os agentes (Robô) são fundamentais para manter a os dados dos de ligações dos clientes atualizados no sistema, atualmente os mesmo estão sendo apontados para o *PostgreSQL*, mas posteriormente quando todos dados da tabela *calls* estiver

migrado para *Elasticsearch*, haverá a mudança no apontamento dos agentes para o *Elasticsearch.*

É importante ressaltar que esta arquitetura proposta é para resolver um problema específico, sendo implementada de forma incremental, podendo haver modificações ao decorrer do projeto, o objetivo principal é propor uma arquitetura em *Big Data* que atenda aos requisitos levantado junto a empresa, sendo um deste principais requisitos o melhoramento dos relatórios, geração de faturas, solucionar o gargalo encontrado na tabela *calls*. Também uma arquitetura que possa trazer nova possibilidade para o negócio, entre estas propostas está a utilização de *Business Intelligence*, que possibilitara entender melhor esta gama de dados, assim podendo proporcionar melhorias estratégicas para cada cliente.

5. Conclusão

O sucesso na construção de uma arquitetura Big Data depende principalmente da clareza do escopo, quais perguntas devem ser respondidas e por consequência o valor a ser gerado, encontrado através dos conhecimentos dos 5Vs, volume, velocidade, variedade, veracidade e valor.

Com um crescimento constante no volume de dados da tabela *calls*, a proposta de uma arquitetura em *Big Data* e de total coerência, uma vez que este modelo arquitetural consta de um poder computacional muito amplo para tratar grande volume e variedade de dados, além de poder oferecer *insight* sobre o valor destes dados que antes não eram percebidos. Oferecendo novas oportunidade de negócios através de analises de *Business Intelligence* aliadas ao *Big Data* e outras tecnologias que fazem parte do ecossistema. O ecossistema utilizando tecnologias com o *Apache Hadoop*, *Hadoop MapReduce*, *Apache HDFS* e o *Elasticsearch* agregam valor fundamental para a arquitetura, a união e a sincronização delas proporciona recursos muito amplos, recursos que atualmente não fazem parte do escopo inicial deste projeto, mais que podem em trabalhos futuros agregar esta arquitetura.

Uma desta tecnologias que vem ser consolidando cada vez mais neste modelo de arquitetura, é o banco *NoSQL*, com um paradigma diferente do banco de dados relacional, eles propõem maneiras diferentes de armazenamento dos dados, seja no modelo de chave-valor, família de colunas, orientado a documentos e baseado em grafos. Cada modelo tem suas particularidades para resolver determinados cenários, mas o modelo relacional também tem suas particularidades, não podemos afirmar que um substitui o outro, mais sim que cada um tem seus benéficos, assim cada um é melhor utilizado para atender determinados cenários.

Para o problema nos relatórios e geração de faturas, constatou-se que o volume de dados é significativamente grande e o seu crescimento também vem sendo significativo. As relações (*joins*) entre a tabela *calls* e outras tabelas tem um custo computacional bastante grande e com isto o tempo de resposta destas consultas vem aumentando juntamente com volume de dados. A utilização do *Elasticsearch* apresentou-se bastante viável, sendo que ele consegue obter resposta com uma performance bem significativa comparada ao modelo relacional, outra benéfico que o *Elasticsearch* proporciona e sendo com um dos seus diferenciais junto ao modelo relacional a utilização em *clusters* de forma distribuídas, podendo assim escalonar esta estrutura de formar vertical, onde se obtém um

ganho computacional aliado com um custo mais baixo. Sendo assim a utilização do *Elasticsearch* uma agregação muito importante para a arquitetura.

Entende-se que o modelo arquitetural proposto ao decorrer do trabalho, consegue satisfazer os requisitos levantados juntamente com a empresa de *Telecom*, não somente os requisitos, mas a viabilidade da implementação de uma solução de *Business Intelligence* que agregara valor significativo ao negócio e a empresa. A arquitetura consegue resolver o problema de gargalo do sistema como também pode proporciona a empresa a entrar em novas possibilidades que antes seria quase inviável.

A arquitetura proposta resolve um problema distinto, levando em consideração os requisitos e o escopo determinado, desta maneira esta solução pode não se aplicar a outros contextos. Para uma possível correlação é necessário ter claramente o contexto do negócio, a estratégia a ser seguida, e por fim uma análise de impacto da arquitetura, para então determinar se o modelo proposto é viável.

Referências

Cordeiro, Leticia. (2016) “Futurecom 2016: TIM oferecerá serviços de Big Data a OTTs”, http://www.bitmag.com.br/2016/10/futurecom-2016-tim-oferecera-servicos-de-big-data-otts.

Cukier, Kenneth. Mayer-Schonberger, Viktor. (2013). “Big data: como extrair volume, variedade e valor da avalanche de informação cotidiana; tradução Paulo Polzonoff Junior”, Rio de Janeiro, Elsevier.

Davenport, Thomas H. (2014). Big Data no Trabalho: Derrubando mitos e descobrindo oportunidades; tradução Cristina Yamagami. 1 ed. Rio de Janeiro, Elsevier.

Friess, Ivan Isaías. (2013). Análise de Bancos de Dados NoSQL e Desenvolvimento de uma Aplicação. Santa Maria: Universidade Federal de Santa Maria (UFSM, RS)

Hanson, Jeff. (2013). “Uma Introdução ao Hadoop Distributed File System”. https://www.ibm.com/developerworks/br/library/wa-introhdfs. Abril.

Henrique, Luis. (2015). “Elasticsearch: realizando buscas no Big Data”. http://www.devmedia.com.br/elasticsearch-realizando-buscas-no-big-data/32180. Abril.

Fowler, Martin; Sadalage, Pramod J. (2013). “NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota Essencial, tradução Acauan Fernandes”, São Paulo, Novatec.

Netto, Adriana Sodré Del Prá; Moro, Evandro Pioli; Ferreira, Fernanda Folly. (2015). “Big Data e suas influências sobre a estratégia das empresas”, http://www.gta.ufrj.br/ensino/eel878/redes1-2015-1/15_1/bigdata/vs.html.

SAS. “O que é Big Data?”. http://www.sas.com/pt_br/insights/big-data/what-is-big-data.html.

Penchikala, Srini. “Big Data com Apache Spark - Parte 1: Introdução”, http://www.infoq.com/br/articles/apache-spark-introduction.

Pensando Grande. (2016). “Empreendedor, descubra como o Big Data pode ajudar a alavancar seus negócios!”, https://blogs.business.microsoft.com/pt-br/2016/11/07/empreendedor-descubra-como-o-big-data-pode-ajudar-alavancar-seus-negocios.

Penteado, Raqueline R. M. et al. “Um Estudo sobre Bancos de Dados em Grafos Nativos”, http://www.inf.ufpr.br/carmem/pub/erbd2014-artigo.pdf.

Santana, Eduardo Felipe Zambom. “Introdução ao Apache Spark”, http://www.devmedia.com.br/introducao-ao-apache-spark/34178.

Taurion, Cezar. (2011). “Conhecendo o Hadoop”. https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/conhecendo_hadoop?lang=en.

Taurion, Cezar. (2012). “Você realmente sabe o que é Big Data?”, https://www.ibm.com/developerworks/mydeveloperworks/blogs/ctaurion/entry/voce_realmente_sabe_o_que_e_big_data?lang=en.

Taurion, Cezar. (2012a). “O caos conceitual e os 5 Vs do Big Data”, http://cio.com.br/opiniao/2012/05/11/o-caos-conceitual-e-os-5-vs-do-big-data/.

Telesintese. (2016) “Vivo cria núcleo de BI e Big Data”, http://www.telesintese.com.br/vivo-cria-nucleo-de-bi-e-big-data.

Van Rijmenam, Mark. “T-Mobile USA Cuts Downs Churn Rate By 50% With Big Data”, https://datafloq.com/read/t-mobile-usa-cuts-downs-churn-rate-with-big-data/512.

Vieira, Marcos Rodrigues. et al. (2012). “*Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data*”, http://data.ime.usp.br/sbbd2012/artigos/pdfs/sbbd_min_01.pdf.