Recapitular o que foi passado até agora.

Usar Slides dos desafios do big data e agora focar em real time analytics

Real time Big Data Analytics

“Usuários finais podem aproveitar das capacidades analíticas cada vez mais sofisticadas através de real-time analytics embarcado em ferramentas de descoberta de dados e aplicações com capacidades de processamento em tempo real e sem necessidade de desenvolvedores intervirem.”

W. Roy Schulte, Analista do Gartner

Real time Big Data Analytics – Desafios

Escalabilidade – Bancos de dados tradicionais não escalam bem.

Schema de dados flexível – Dificuldades de manipulação de fontes de dados heterogêneas e dados não estruturados.

Tempo de ingestão e preparo dos dados – Sistemas tradicionais operam com janelas de tempo pré-definidas em dia-1 ou n-horas de atraso.

Casos de uso – Realtime Big Data Analytics

Na Black Friday de 2016, a Sprint presenciou uma baixa significativa no acesso ao sprint.com. Anteriormente, teria levado horas para investigar manualmente os dados para identificar a causa raiz e consertar o problema. Ao invés disso, o time de TI foi alertado imediatamente, identificando o problema e rapidamente corrigindo a falha.

Esse tipo de agilidade é crucial para a empresa vendar mais celulares.

3 bilhões de eventos por dia de logs, bancos de dados, e-mails, logs de sistema, mensagem de texto e APIs internas e externas.

<https://www.elastic.co/use-cases/sprint>

ELK Stack

O que é :

Uma coleção de três produtos open source – Elasticsearch, Logstash e Kibana – mantidos pela Elastic.

Elasticsearch : NoSQL database baseado no Apache Lucene (motor de busca)

Logstash : ferramenta de pipeline de dados com conexão a vários tipos de fontes, executando transformações e exportando dados para vários destinos.

Kibana : Camada de visualização que funciona sobre o Elasticsearch

<https://logz.io/wp-content/uploads/2018/08/image21-1024x328.png>

Produtos adicionais – Canvas

Ferramenta de visualização com capacidade de ajuste fino via CSS e criação de novos componentes

<https://www.elastic.co/assets/blte82ce6dcd493cdb4/business-analytics-canvas-animation.gif>

ELK - Benefícios

* Open Source,
* Comunidade ativa,
* Altamente escalável e distribuído,
* Schemaless data model,
* APIs de fácil acesso

Utilização

Social media Analytics

<https://www>.elastic.co/assets/blte771400b1f9a444b/business-analytics-sentiment-analysis.png

Serviços Financeiros

<https://www>.elastic.co/assets/bltf73b414d490af1f0/business-analytics-financial-services.png

Varejo

<https://www>.elastic.co/assets/bltea8c5544a2e677e3/business-analytics-retail-ops.png

Web Analytics

<https://www.elastic.co/assets/blt9da3b921073c9dae/business-analytics-web-analytics.png>

ELK – o que não é

Não é banco de dados – armazenamento temporário de dados por n-dias

Não há garantia de consistência (ACID)

Otimizado para consulta, não alterações

Outras opções

Logz.io

Splunk

MongoDB + aplicação de visualização

Bancos de dados em memória (ex.: SAP Hana)

Trabalho prático – Dados de eventos/Sensor

Dados : <http://www.datapoa.com.br/dataset/poatransporte>

Dados sobre transporte público via API REST + enriquecimento

Dados de sensores (simulação) – REAL TIME

Elastic Conceitos

[https://cdn-images-1.medium.com/max/800/1\*0gUMUCd81Oxu-npn-NZcTw.png](https://cdn-images-1.medium.com/max/800/1*0gUMUCd81Oxu-npn-NZcTw.png)

<https://www.timroes.de/2015/02/07/kibana-4-tutorial-part-3-visualize/>

Aggregations

Kibana trabalha com dois tipos de agregações :

**Bucket** : Date Histogram, Histogram, Range, Terms, Filters, Significant Terms, Geohash

**Metric** : Count, Average/Sum, Max/Min, Unique Count, Percentiles

Bucket Aggregations

Esse tipo de agregação agrupa todos os documentos em vários “buckets”, cada um contendo um subconjunto dos documentos indexados. A decisão de qual bucket um documento específico vai ser colocado pode ser baseado no valor de um campo, um filtro customizado ou outros parâmetros.

**Date Histogram**

A agregação “Date Histogram” requer um campo de tipo data e um intervalo. Cada documento vai ser colocado em um bucket que o campo de data está dentro do intervalo.

Ex.: datapoa - Passageiros por minuto – Neste caso haverá um bucket para cada minuto e cada um vai ter documentos onde o campo data está dentro do intervalo.

**Histogram**

A agregação “Histogram” é praticamente igual a “Date Histogram”, exceto que é possível utilizar com qualquer campo numérico. Para isso, basta escolher um campo número e um intervalo (que neste caso é um número). A agregação vai construir um bucket para cada intervalo e colocar os documentos cujo valor estão dentro do intervalo.

Ex.: Análise de logs : um histograma usando o campo UTC timezone offset (diferença de tempo do timezone com UTC em segundos) com um intervalo de 3600 vai criar buckets para cada timezone (UTC+1, UTC+2 etc) e colocar todos os documentos que pertencem ao intervalo no seu respectivo bucket.

**Range**

Esse tipo de agregação funciona como um histograma manual. É necessário especificar um campo do tipo número, mas cada intervalo deve ser configurado manualmente.

Ex.: datapoa – Criar uma a agregação do tipo range sobre a quantidade de passageiros de 0-10, 10-30 e 30-\*.

**30-\*** : significa um intervalo aberto (maior que 30)

**Terms**

Esse tipo de agregação cria buckets de acordo com o valor de um campo. É similar a cláusula GROUP BY do SQL. Um campo de qualquer tipo é especificado e isso vai criar um bucket para cada valor existente naquele campo.

Ex.: datapoa – Criação de uma agregação por empresa, os buckets serão cada um dos nomes de empresa presentes nos documentos.

**Filter**

A agregação do tipo filtro é completamente flexível. Só é necessário definir um filtro para cada bucket, e cada documento que corresponder ao filtro vai ser colocado no bucket. O filtro especificado para cada um dos buckets pode ser qualquer coisa, logo não é necessário haver uma relação entre eles.

Ex.: twitter .: criar uma agregação de filtro com uma consulta do tipo "place.country\_code:(tr or jp)" e um segundo filtro “user.follower\_count:[1000 TO \*]”. Dessa forma a agregação vai criar dois buckets, um contendo os tweets da Turquia e Japão, e outro com os tweets dos usuários que possuem 1000 ou mais seguidores.

**Significant terms**

Esse tipo de agregação pode ser usada para encontrar termos “incomumente comuns” em um conjunto de documentos.

Dado um subconjunto, a agregação encontra todos os termos que aparecem nesse subset mais frequentemente que poderia ser esperado no conjunto total de documentos.

A agregação Terms é utilizada para identificar características do conjunto de dados enquanto Significant Terms é utilizado para reconhecer suas características especiais.

Ex.: Busca – sugerir H5N1 quando um usuário pesquisa por “gripe aviária” : Nesse caso o termo selecionado não é simplesmente o mais popular no conjunto. É um termo que sofreu uma mudança significativa na popularidade medida entre o conjunto de dados **foreground** e **background**. Se o termo “H5N1” existe somente em 5 documentos em um índice de 10 milhões, mas é encontrado em 4 de 100 documentos que compõem o resultado da busca, isso provavelmente é relevante. 5/10.000.000 contra 4/100 é uma grande mudança na frequência.

**Geohash**

O Elasticsearch cria buckets para coordenadas geográficas próximas umas das outras.

Metric Aggregations

Após agregar os dados em buckets, cada documento será atribuído a um bucket específico. Agora, podemos especificar Agregações de Métricas para calcular um valor único para cada bucket.

Nas visualizações, a agregação de buckets geralmente vai ser usada para determinar a “primeira dimensão” do gráfico (ex.: para um gráfico de pizza, cada bucket é uma fatia do gráfico, para um gráfico de barras cada bucket vai ter sua própria barra). O valor calculado pela agregação de métricas será mostrada como uma “segunda dimensão” (ex.: no gráfico de pizza, o percentual relacionado ao círculo, no gráfico de barras a altura da barra no eixo Y)

**Count** : Não se trata realmente de uma agregação, mas simplesmente o total de documentos em cada bucket.

**Average/Sum** : Para agregações de média e soma é necessário especificar um campo numérico. O resultado para cada bucket será a soma/média daquele campo.

**Max/Min** : Da mesma forma que a média e soma, essa agregação requer o uso de um campo numérico. Ela vai retornar o valor máximo/mínimo que pode ser encontrado nos documentos presentes naquele bucket.

**Unique count** : A contagem única retorna quantos valores diferentes/únicos existem para um determinado campo para um bucket.

**Top hits aggregation** : Basicamente uma métrica de agrupamento. É utilizado para agrupar certos itens baseados em uma característica, mas dentro desse grupo se deseja extrair o melhor resultado.

Ex.: Um índice contém todas as músicas da biblioteca de músicas do iTunes de um usuário. Primeiro, procuramos músicas por gênero. O resultado será a contagem de documentos para cada gênero (rock, pop, samba, etc). Agora, se quisermos pesquisar pela palavra “love” no título da música por gênero, mas queremos apenas o “best hit” por cada gênero, utilizamos uma métrica do tipo **Top hits.**

**Percentiles :** a agregação de percentil é um pouco diferente uma vez que ela não vai resultar em um único valor por bucket, mas sim em múltiplos valores. Quando especificado uma agregação por percentil é requerido um campo numérico e múltiplos valores percentuais. O resultado da agregação será o valor para o qual o percentual de documentos estará dentro desta faixa de valor (são menores que esse valor).

Ex.: Twitter – utilizando uma agregação por percentil no campo user.followers\_count e especificando os percentis 1, 50 e 99, o resultado será três valores agregados para cada bucket. Vamos assumir que só existe um bucket e o 1º percentil tem o valor 7. Isso quer dizer que 1% de todos os tweets desse bucket tem uma contagem de seguidores de 7 ou menos. O resultado do 50º percentil é 276, significando que todos os tweets desse bucket tem uma contagem de seguidores de 276 ou menos. O percentil 99 tem um valor de 17000, significando que 99% dos tweets no bucket tem uma contagem de seguidores de 17000 ou menor.

1. Instalação do Elasticsearch e Kibana
   1. <https://www.elastic.co/downloads/elasticsearch>
   2. <https://www.elastic.co/downloads/kibana>
   3. Pré requisitos
      1. Java instalado (java –version)
      2. JAVA\_HOME configurado
2. Configuração das ferramentas
   1. Elasticsearch.yml
   2. Kibana.yml
3. Com o Elastic Executando
   1. Criação de índices
   2. Mappings
   3. Ingestão de dados via python

Timelion : <https://www.elastic.co/blog/timelion-tutorial-from-zero-to-hero>

.es(q=empresa:MAIS,index=datapoa\_carros,timefield=datahora).bars(), .es(q=empresa:CARRIS,index=datapoa\_carros,timefield=datahora),

.es(index=datapoa\_carros,timefield=datahora,split=empresa:2)

.es(index=datapoa\_carros,timefield=datahora,split=empresa:5,metric=sum:total\_passageiros)

.es(index=datapoa\_carros,timefield=datahora,q=empresa:MAIS,metric=sum:total\_passageiros).color(blue),.es(index=datapoa\_carros,timefield=datahora,q=empresa:CARRIS,metric=sum:total\_passageiros).color(green),.es(index=datapoa\_carros,timefield=datahora,metric=avg:total\_passageiros).label('Media passageiros').color(red)

.es(index=datapoa\_carros,timefield=datahora,q=empresa:MAIS,metric=sum:total\_passageiros).color(blue).bars(),.es(index=datapoa\_carros,timefield=datahora,q=empresa:CARRIS,metric=sum:total\_passageiros).color(green).bars(),.es(index=datapoa\_carros,timefield=datahora,q=\*,metric=avg:total\_passageiros).trend()