FACULDADE DE TECNOLOGIA DE SÃO PAULO

FÁBIO MASSAHIRO AKAMINE

Data Stream em Tempo Real

SÃO PAULO
2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

FÁBIO MASSAHIRO AKAMINE

Data Stream em Tempo Real

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.
Orientador: Prof. Valter Yogui.

SÃO PAULO
2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

FÁBIO MASSAHIRO AKAMINE

Data Stream em Tempo Real

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Parecer do Professor Orientador
aprovado

Conceito/Nota Final: 9,0

Atesto o conteúdo contido na postagem do ambiente TEAMS pelo aluno e assinada por mim para avaliação do TCC.

Orientador: Prof. Valter Yogui

SÃO PAULO, 18 de junho de 2021.

Assinatura do Orientador

Assinatura do aluno

Dedico este trabalho a meus pais Edson e Silvia, a minha família, a meus amigos e meus professores, em especial ao meu orientador Prof. Valter Yogui.

AGRADECIMENTOS

A maior palavra é gratidão.

Gratidão À minha família que sempre me apoiou em várias jornadas.

Gratidão aos amigos que compartilharam as aulas, as dores, felicidades e os momentos únicos.

Gratidão a Fatec e seus professores que dividiram experiências, conhecimento e seu tempo.

”Não podemos analisar o que não podemos medir.”

(W. Edwards Deming e Peter Drucker)

Lista de Ilustrações

RESUMO

A evolução da Informática fez a produção de dados crescer de forma exponencial. Nunca na história humana tantos dados foram criados, armazenados e compartilhados, gerando novos conhecimentos, tecnologias e oportunidades. Além dos dados gerados pela interação humana, a automatização gerou um novo fluxo de dados, a comunicação entre máquinas usando robôs e aplicativos que trocam mensagens sem interferência humana. Junto a esses fatores, a velocidade crescente da *internet* exige soluções em tempo real para suprir as necessidades de pessoas e empresas. Para suportar esse contexto, um novo tipo de processamento foi desenvolvido, o *Data Stream* em Tempo Real. Por ser uma tecnologia recente, não existe uma padronização dos sistemas ou um modelo amplamente aceito. O presente trabalho analisa diversos trabalhos acadêmicos e casos reais focados no *Data Stream* em tempo real.

Palavras-Chave: *Data Stream*; Tempo Real; Datasets Infinitos;

ABSTRACT

The evolution of Informatics has made data production grow exponentially. Never in human history has so many data been created, stored and shared, generating new knowledge, technologies and opportunities. In addition to the data generated by human interaction, automation has generated a new flow of data, communication between machines using robots and applications that exchange messages without human interference. Along with these factors, the increasing speed of the internet requires real-time solutions to meet the needs of people and companies. To support this context, a new type of processing was developed, the Real Time Data Stream. As it is a recent technology, there is no standardization of systems or a widely accepted model.The present work analyzes several academic works and real cases focused on the Data Stream in real time.

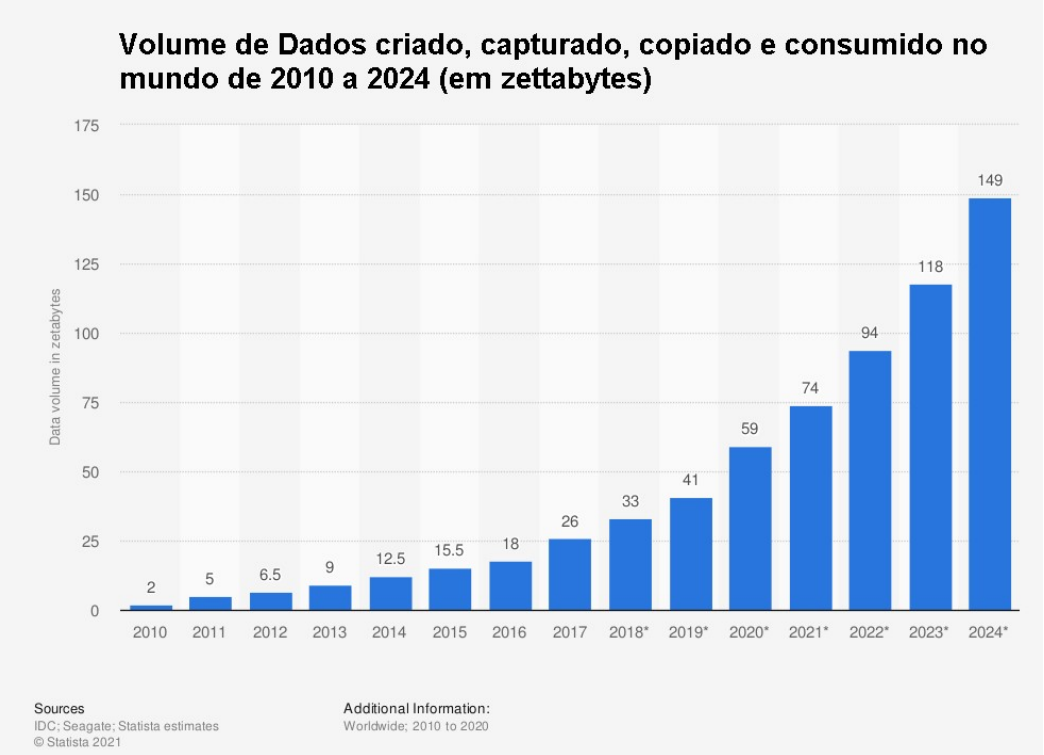
Keywords: *Data Stream*; Real Time; Infinite Dataset;

SUMÁRIO

1 INTRODUÇÃO

A evolução da tecnologia da informação é marcada pela captura e transmissão crescente de dados para aplicações em produtos, serviços e pesquisas. Por dia, 4.2 bilhões de usuários usam mídias sociais (Johnson, 2021), enquanto o Grande Colisor de Hádrons gera um *petabyte* de informação por dia e seu sucessor, o Futuro Colisor Circular, tem previsão de gerar na ordem de *hexabytes* (CERN, 2017).

Figura 1 - Volume de Dados criado, copiado e consumido no mundo de 2010 a 2024 (em *zettabytes*)



Fonte: Adaptado de Holst (2020)

Esses dados impõem desafios, seja pelo curto tempo de vida útil da informação ou pela incapacidade de armazenar de forma econômica a quantia de dados gerada para processamento. Nesse contexto, utilizar um sistema de *Data Stream* em Tempo Real permite extrair os valores dos dados com baixa latência e guardar apenas as informações relevantes, reduzindo a necessidade de armazenamento.

O processamento *Data Stream* é uma evolução natural da procura por arquiteturas robustas de processamento de grandes massas de dados, que teve

origem com o recenseamento nos Estados Unidos usando a Máquina Tabuladora e ganhou cada vez mais usos em setores financeiros, industriais e gerenciais com novas tecnologias ampliando a capacidade de captura de dados e novas formas de extrair valor como as *Data Warehouses*, o *Big Data* e a tecnologia *Internet of Things* (IoT).

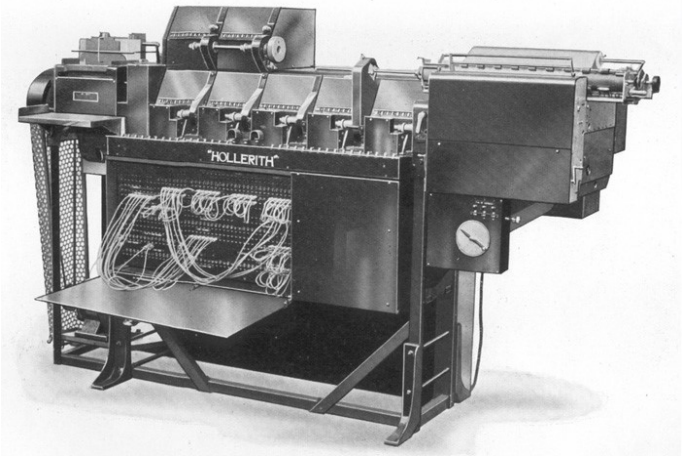
2 METÓDOS DE PESQUISA

O presente estudo foi feito com base em documentos técnicos relacionados ao tema *Data Stream,* disponibilizados pelos *sites* agregadores de pesquisa *Springer Link* e Academia, nas ferramentas *Google MapReduce*, *Hadoop*, *Apache Storm* e no Modelo Teórico *Dataflow*.

3 A MÁQUINA TABULADORA

Em 1890 foi executado pelo governo estadunidense o primeiro processamento de dados em massa com o uso da Máquina Tabuladora de Herman Hollerith. O censo de 1880 demandou cerca de sete anos de análise, enquanto com a nova tecnologia foram necessários apenas seis meses (CRUZ, 2001).

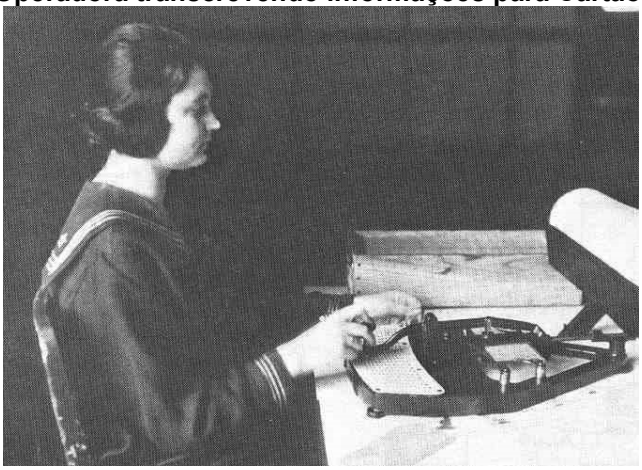
Figura 2- Máquina Tabuladora de Herman Hollerith



Fonte: Extraído do *site* da Universidade de Columbia (1932)

Hollerith utilizou conceitos como o cartão perfurado do inventor francês Joseph Marie Jacquard e a programação binária do filósofo Francis Bacon para produzir sua máquina, que provou ser capaz de realizar as tarefas de forma confiável e eficiente, mas o uso de cartões perfurados transcritos manualmente por operadores era um limitante para sua capacidade.

Figura 3 - Operadora transcrevendo informações para Cartão Perfurado



Fonte: Extraído do *site* da Universidade de Columbia (1920)

O desenvolvimento de novos sensores e meios de armazenamento de dados sem interferência humana aumentou a velocidade de coleta e a confiabilidade dos dados, mas desafiou os limitados equipamentos da época, exigindo uma abordagem diferente de processamento com métodos e padrões confiáveis para manipular os dados.

4 O PROCESSAMENTO *BATCH*

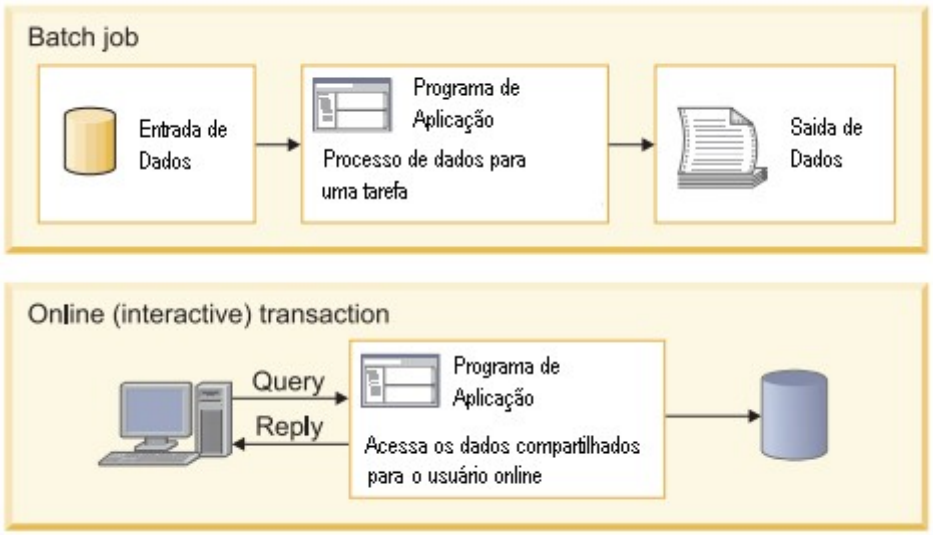
O processamento *Batch* surgiu com as demandas por um processamento de dados seguro, confiável e a prova de falhas. A busca por uma arquitetura para este processo culminou em um novo equipamento, o *Mainframe*. A arquitetura do *Mainframe* se dedica a processar milhares de registros de forma segura com o processamento simultâneo.

Como proposto pela *Internacional Business Machine* (IBM): “Um *Mainframe* é aquilo que empresas usam para sediar seus bancos de dados comerciais, servidores de transações, e aplicações que requerem um grande nível de segurança e disponibilidade que não é possível encontrar em uma máquina menor” (MAINFRAME, 2008, p. 05).

No início, as atividades eram registradas durante o dia e o *Mainframe* processava durante o período noturno, pois dependia de dados que não fossem alterados durante a execução do sistema.

O advento de meios de pagamento como cartões e transações *online* exigiu uma solução para processar as operações mesmo durante a execução de programas. Com isso foram implantado dois modos de processamento: o *Batch* e o *Online*.

Figura 4 - O Processamento *Batch* e o Processamento *Online*



Fonte: Adaptado de Mainframe (2008)

O **Processamento *Batch*** *é* realizado sem a interferência do usuário, onde os programas são listados no computador que realiza o acesso ao banco de dados e executa as instruções de forma sequencial. As principais características de um processamento *Batch* são:

a) Grandes quantidades de dados de entrada são processados e armazenados, com grande acesso ao banco de dados e geração de arquivos de saída;
b) Uso de controle de processamento agendado para processar, de forma a otimizar o trabalho.

Já no **Processamento *Online*** *a*s interações entre o usuário e o sistema são muito rápidas, feitas de pequenas interações com respostas imediatas. Exemplos de operações podem ser vistas em caixas eletrônicos ou em transações feitas com cartões de crédito e débito. O processamento é delimitado por períodos de disponibilidade (24 horas, seis dias por semana, após 17h00, etc,) para o *Batch* e o *Online*.

A arquitetura do *Mainframe* se baseia em dividir o sistema em pequenos subsistemas chamados de partições lógicas (*Logical Partitions*), controladas por um programa supervisor (*Hypervisor*) onde os processos são executados de forma paralela.

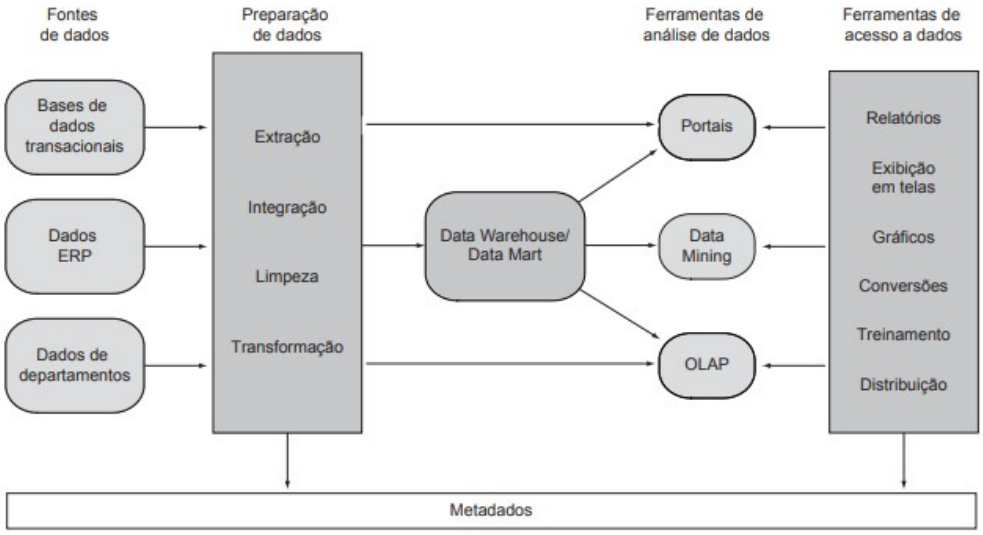
Apesar da sua confiabilidade, o alto valor para programar e manter os equipamentos *Mainframe* são limitantes ao seu uso e manutenção como sistema principal de uma empresa, além de ser um sistema não escalável, exigindo conhecimento especializado para executar em máquinas separadas ou compra de máquinas maiores, bem como não ser compatível com novas tecnologias tais como computação e armazenamento em nuvem.

5 *DATA WAREHOUSE* E B.I

O processamento de dados trouxe uma grande quantidade de informações que eram registradas em tabelas de bancos de dados separados, e não havia ferramentas adequadas para cruzá-los e extrair seu valor. Então, um novo conceito começou a se estabelecer, a busca pela centralização e integração dos dados da empresa para produzir relatórios cada vez mais precisos e inteligentes produziu as *Data Warehouses*, conceituado por Inmon (2007, p. 01) como “coleção de dados orientados ao assunto, não volátil, integrada e variável no tempo para apoiar as decisões da administração”.

Com isso, os dados deixam de ser armazenados em diversos repositórios de informações para ganhar uma arquitetura que se preocupa em garantir a qualidade, consistência, centralização e capacidade de relacionar dados de diferentes fontes ao orientar a assuntos.

Figura 5 - Fluxograma de uma *Data Warehouse*



Fonte: Adaptado de Mattioda e Favaretto (2009)

Uma *Data Warehouse* típica possui um banco de dados relacional, uma solução do tipo Extração, Carregamento e Transformação (ELT) que prepara os dados para análise, uso de ferramentas estatísticas e mineração de dados para relatórios. Mesmo com toda a capacidade de uma *Data Warehouse*, a dificuldade em escalar a solução para atender empresas de caráter global - que gera grande

quantidade de informações em múltiplos países - e o desafio de armazenar toda a produção de dados de forma segura e econômica limitam sua aplicação.

6 *INTERNET OF THINGS* E UM MUNDO INTERCONECTADO

Em 1999, Kevin Ashton promoveu a tecnologia Identificação por Radiofrequência (RFID) aplicada em Logística e usou o nome "*Internet* das Coisas" para chamar a atenção dos executivos (LUETH, 2014). Apesar de essa ser a primeira menção ao nome, o termo não era válido, pois não existia interação entre os objetos e a *internet*.

Nos anos seguintes, a miniaturização de componentes eletrônicos e a redução de seus custos permitiram imbuir sensores e processadores em equipamentos com intuito de aumentar a eficiência, usando o processamento para controlar funções básicas, monitorar parâmetros, coletar dados e gerar relatórios de erros, criando o conceito de tecnologia embarcada.

Com a popularização da *internet*, o interesse em usar a capacidade já existente da tecnologia embarcada e conectar via rede com um grande centro para controle e monitoramento produziu protocolos e tecnologias de comunicação tipo Máquina-Máquina.

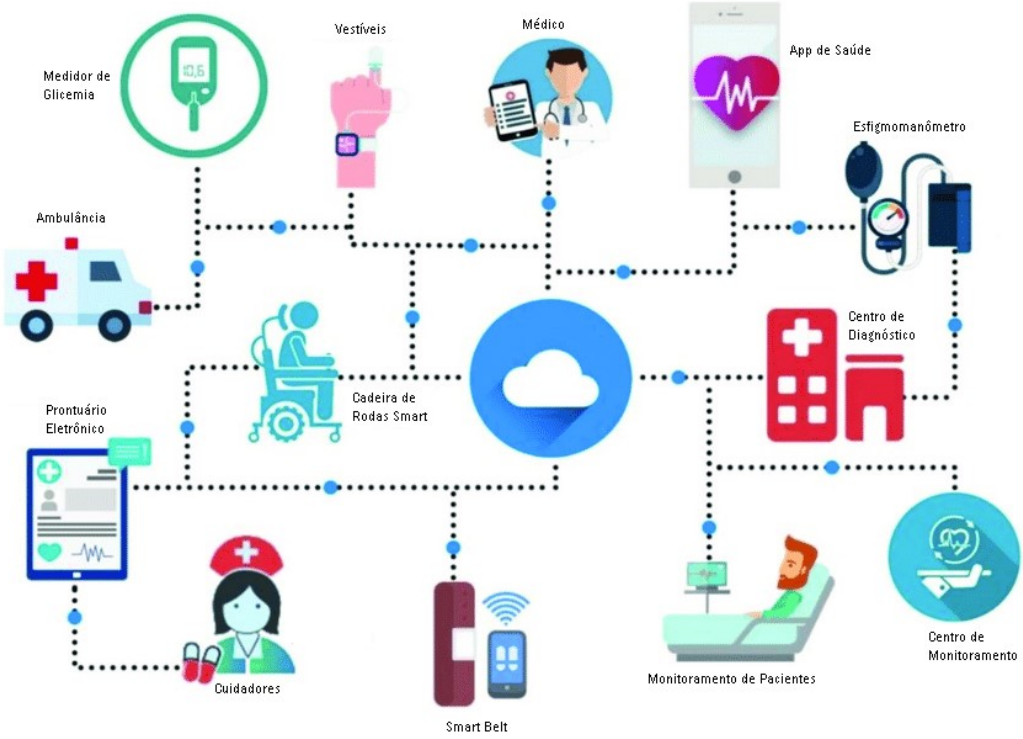
No uso doméstico para esta tecnologia, a ampla adoção de redes *Wi-Fi* em casas fez as empresas voltarem os olhos para aplicações de casas inteligentes por meio da conexão de aparelhos a *internet* e gerou uma definição mais precisa da tecnologia IoT, como "uma infraestrutura global para a sociedade da informação, permitindo serviços avançados interconectando coisas (físicas e virtuais) com base em tecnologias de informação e comunicação interoperáveis existentes e em evolução" (GENEVA, 2012, p. 01).

A evolução natural se deu com o uso de Assistentes Virtuais capazes de se comunicar e controlar outros equipamentos, criando funcionalidades e abrindo o caminho para conceitos ainda maiores como:

a) *Internet* das Coisas Médicas: monitoramento em tempo real de equipamentos de pacientes em Unidades de Tratamento Intensivo (UTI) ou de pacientes em suas residências, com prevenção de falhas;
b) *Internet* das Coisas Industriais: criação de modelos virtuais de fábricas com dados gerados pelos equipamentos e sensores para aplicar tecnologias preventivas e um controle de qualidade rigoroso;

c) *Internet* das Coisas Automotivas: uso de dispositivos para carros autônomos e controle de trânsito;

Figura 6 - *Internet* das Coisas Médicas



Fonte: Adaptado de TOOR; USMAN; YOUNAS; FONG; KHAN; FONG (2020)

d) Casas Inteligentes: plataformas de automação inteligente como *Alexa*, *Google Home* e *Apple HomeKit* se popularizaram e fornecem soluções baratas e simples;

e) Cidades Inteligentes: a expansão urbana cria grande pressão sobre a capacidade de gerir os recursos de forma sustentável e o uso de sensores inteligentes é proposto para monitoramento de índices de consumo de energia, níveis de poluição e outros indicadores em tempo real;

f) Agricultura Inteligente: uso de sensores no solo para monitorar níveis de nutrientes, umidade e temperatura.

Apesar do grande leque de aplicações, alguns desafios impedem a adoção imediata da tecnologia em larga escala, como por exemplo:

a) Grande número de objetos de redes distintas com diferentes capacidades de transmissão de dados e disponibilidade de rede, usando uma profusão de protocolos;

b) Vários dispositivos podem ter grandes picos de produção de dados que se assemelham a ataques do tipo *Denial of Service* (DOS) que precisam ser suportados pela rede;
c) Os dispositivos podem ter uma hierarquia de prioridades (aparelhos médicos de uma enfermaria, berçário e UTI) transmitidos em rede comum, exigindo flexibilidade para priorizar os dados importantes;
d) Capacidade de verificar se os dispositivos estão saudáveis, enviando dados corrompidos ou com problemas, não apenas monitorar, mas agir de forma preventiva;
e) Pela natureza de grande volume, pequenos *bugs* de dispositivos que ocorrem apenas uma vez por um milhão de ciclos se torna uma ocorrência comum e deve ser tratada;
f) Como os dispositivos podem ficar desconectados da rede por longos períodos, os dados acumulados podem ser enviados em grandes mensagens e o sistema precisa processá-los sem alterar sua latência para os outros dados;
g) Pela natureza dos fenômenos monitorados em tempo real, os dados são gerados em um fluxo constante e devem ser processados com a menor latência possível para preservar sua relevância, não sendo possível armazenar para processamento.

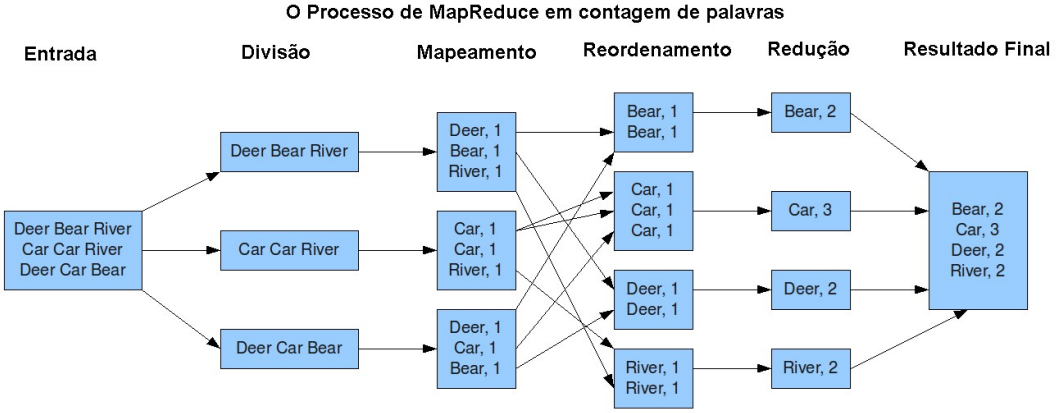
7 *BIG DATA*

A expansão da tecnologia exigiu o aumento dos bancos de dados para comportar grandes volumes a baixo custo. Os dados se tornaram abundantes e sua coleta mais rápida, gerando intensos fluxos a serem armazenados e processados, complexos demais para os métodos tradicionais.

Surge, então, o termo *Big Data*, um conceito baseado inicialmente em “três Vs”, **volume** (quantidades extremamente grandes de dados), **velocidade** (criados em tempo real) e **variedade** (dados estruturados, semiestruturados e não-estruturados) e depois acrescentados mais três “Vs”, a *veracidade* (os dados podem ser desorganizados e com erros), o *valor* (os dados contém informações) e a *variabilidade* (os dados podem estar em constante mudança de acordo com contexto em que são gerados) (KITCHIN; MCARDLE, 2016).

Uma das primeiras ferramentas para lidar com a grande quantidade de dados foi o *Google MapReduce*, um modelo de programação distribuído para processamento em paralelo usando um *cluster* de computadores, um sistema de arquivo distribuído (*Google File System/G*FS) e um banco de dados *BigTable* (NoSQL).

Figura 7 - Funcionamento Geral do Processo de MapReduce



Fonte: Adaptado de *TodaySoftMag* (2015)

No processo do *MapReduce*, os dados são divididos em blocos e passam pelo processo *Map*, que mapeia criando duplas de chave/dados e depois reduz agrupando pelas chaves. Um sistema organizador administra o processamento e

gravação do arquivo em diferentes máquinas, garantindo a integridade ao usar cópias redundantes em diferentes discos do *cluster*.

A distribuição em *cluster* de computadores permite que o sistema grave e retorne os dados de forma mais rápida que um único servidor, distribuindo o trabalho e permitindo uso de máquinas de menor capacidade com escalabilidade.

Apesar do grande avanço na solução de armazenamento, ela não é compatível com tecnologias como *Machine Learning*. Para funcionar corretamente, um sistema *Machine Learning* precisa ser capaz de analisar todo o *dataset* de uma única vez, mas a natureza do *Big Data* obriga a particionar em *subdatasets* que geram resultados parciais a serem mesclados. Este processamento não garante que todas as tendências foram analisadas, pois elas podem estar difundidas entre os *subdatasets*.

Além disso, o sistema *Google MapReduce* usa um sistema de arquivos GFS que grava os dados no disco para manipulá-los, tornando os acessos lentos comparado ao uso do processamento em memória do processo *Data Stream*.

8 FLUXO DE DADOS *DATA STREAM*

O avanço na tecnologia trouxe um novo tipo de fluxo de dados, os alimentadores de dados automatizados. Essas fontes de dados são caracterizadas por serem infinitas, observando que tais dados são transmitidos em alta velocidade e gerados por distribuições não estáticas em ambientes dinâmicos (GAMA; GABER, 2007).

Um novo conceito é introduzido para este tipo de fluxo, o *Data Stream*, definido por Guha et al (2003 p. 01) como “uma sequência ordenada de instâncias que podem ser lidas apenas uma vez ou um número pequeno de vezes”. Segundo Babcock et al (2002), se enquadram nessa categoria os dados gerados por sensores e a comunicação Não Humana (*Non-Human Traffic*), que possuem características diferentes dos fluxos normais, quais sejam:

- Os elementos do *Stream* chegam *online*;
- O sistema não possui controle sobre a ordem em que os elementos chegam;
- Os *Data Stream*s são de tamanho infinitos (*unbounded*);
- Uma vez que o elemento do *Data Stream* seja processado, ele é descartado ou arquivado, não podendo ser recuperado na memória exceto se explicitamente armazenado, sendo a memória menor que o tamanho total do *Data Stream*.

Os sistemas *Batch* foram tradicionalmente usados para processá-los, com as seguintes limitações:

- A formação da captura de todos os dados do lote completo para processamento é de alto custo devido ao grande volume de dados;
- No caso da escolha de amostras de dados para processamento ao invés da análise do fluxo total, o sistema perde a exatidão, pois não é possível garantir que todos os dados de interesse estão dentro do lote;
- Em processamentos *Batch*, existe a incerteza de que o lote esteja completo pela natureza desorganizada dos dados e não é possível determinar se um lote está completo devido à incerteza sobre o tempo em que os dados foram ingeridos pelo sistema;
- A latência do sistema é alta, com alternância de grande concentração de serviço em curtos períodos (tempo de processamento) e períodos de inatividade (captura de dados para formação do lote);

- Alto custo de máquinas especializadas para o processamento e conhecimento técnico.

A necessidade de mover o foco de tomadas de decisões baseados em dados históricos para a análise e projeção de dados futuros não é atendida por sistemas *Batch*. O uso de sistemas computacionais capazes de realizar projeções estatísticas e hipóteses move o foco de respostas exatas para o uso de aproximações, pois a análise de tendências e o monitoramento de ocorrências são exemplos de informações onde o uso de aproximações em tempo real são melhores do que resultados exatos com grande latência. Nestes casos, é preciso uma arquitetura capaz de processar os diferentes tipos de *Data Stream* de acordo com suas propriedades intrínsecas e o tipo de informação a ser obtida.

9 SISTEMAS PARA PROCESSAMENTO *DATA STREAM* EM TEMPO REAL

Para processar o *Data Stream*, diversas implementações inspiradas no sistema *Hadoop* foram propostas se baseando em dois tipos:

a) **Processamento de Micro-lotes,** que captura os dados para sua divisão em micro-lotes para processamento e armazenamento da informação;
b) **Processamento de Streaming,** no qual os dados são recebidos e processados em tempo real e o resultado anterior é reprocessado e atualizado com os dados atuais.

As implantações desses modelos, criados por comunidades e empresas diversas, resultam em ferramentas que possuem vantagens e desvantagens distintas. Com a falta de um consenso sobre um funcionamento geral, foi proposto o modelo *Dataflow* (Akidau et al, 2015), que será aqui estudado e se baseia nas seguintes características:

- Conceito de *Data Streaming*, definido por Akidau et al (2015, p. 01) como “um tipo de *engine* de processamento de dados projetado para *datasets* infinitos”. Um sistema ideal deve ser capaz de:
- Permitir o cálculo de resultados ordenados de eventos no tempo, ordenados em janelas e ordenados por características dos próprios dados;
- Separar a noção lógica de lotes, micro-lotes e *streaming*;
- Possuir implantações em sua *pipeline* relacionadas a quatro dimensões citadas abaixo, garantindo clareza, composição e flexibilidade, englobando:
  - o que está sendo computado;
  - onde o evento está sendo computado;
  - onde no tempo de processamento o evento está se materializando;
  - como o evento presente se relaciona com os eventos passados.

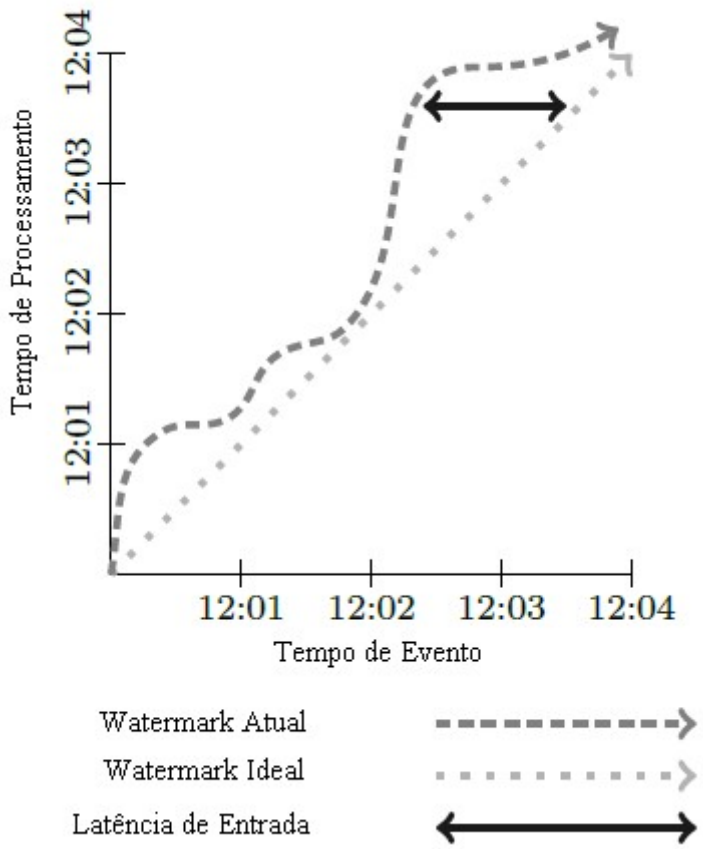
Esses requisitos podem ser alcançados por meio da adoção de alguns preceitos fundamentais:

a) Um **Modelo de Janela**, que suporte janelas de eventos sem alinhamento e uma API simples para sua criação e uso;
b) Um **Modelo de Gatilho** que vincule a saída do processamento com as características do pipeline para criar os gatilhos pela semântica;

c) Um **Modelo de Processamento Incremental,** que suporte atualizações e reprocessamentos nos modelos acima.

Pela sua natureza, os dados de um *Data Stream* possuem uma característica: a incerteza sobre a ingestão de todos os dados em um determinado período.

Figura 8 - Tempo de Evento Ideal vs Tempo de Ingestão



Fonte: Adaptado de Akidau et al (2015)

Na Figura 8, a diferença causada pelo atraso na geração e ingestão dos dados causa a Latência de Entrada é medida pela diferença entre *Watermarks* (Marca de Tempo que registra quando o arquivo foi gerado e outra para quando foi ingerido pelo sistema). Este fenômeno impede que seja possível afirmar que todos os dados de uma janela de evento foram recebidos e que dados sejam recebidos fora da sua respectiva janela de tempo.

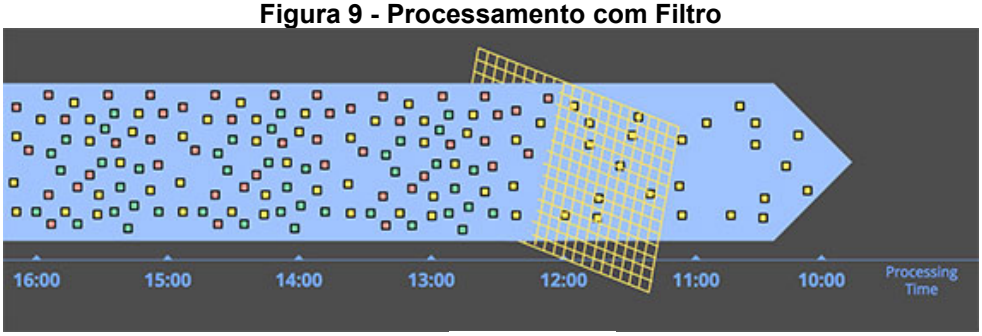
Outro problema gerado é quando ocorrem eventos pareados do tipo início-fim, nos quais a sinalização de evento-início ou evento-fim é recebida fora da janela do seu respectivo par ou não é recebida devido a problemas na rede. Para solucionar

este problema, foram sugeridos quatro tipos de janela de processamento, os quais serão abordados a seguir.

9.1 Processamento por filtro

Tipo de processamento *streaming* onde os dados são ingeridos e imediatamente processados e filtrados. Apenas o resultado do processo é armazenado, sendo indicado para processamento de palavras-chaves e outros eventos independentes de tempo (Agnósticos ao Tempo).

Figura 9 - Processamento com Filtro

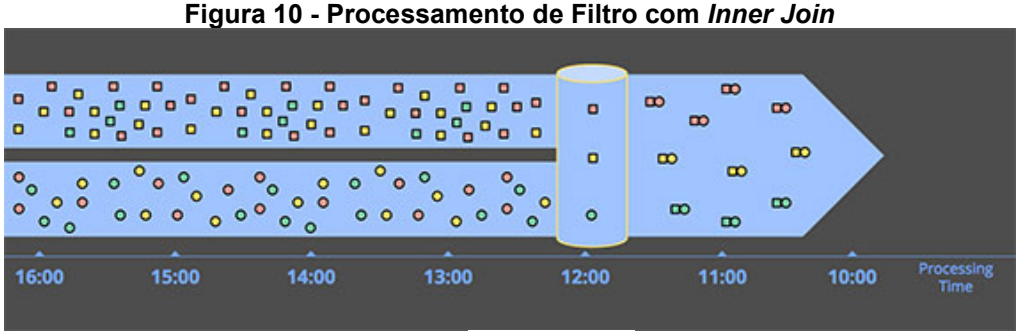


Fonte: Akidau (2015)

9.2 Processamento de filtro com *Inner Join*

Para dados que sejam pareados, como dados do tipo Início-Fim de sessão, os dados podem ser persistidos em memória até que seu par correspondente seja ingerido, com uma política de coleta de dados não-pareados para sua posterior análise ou exclusão com base em um parâmetro de tempo.

Figura 10 - Processamento de Filtro com *Inner Join*

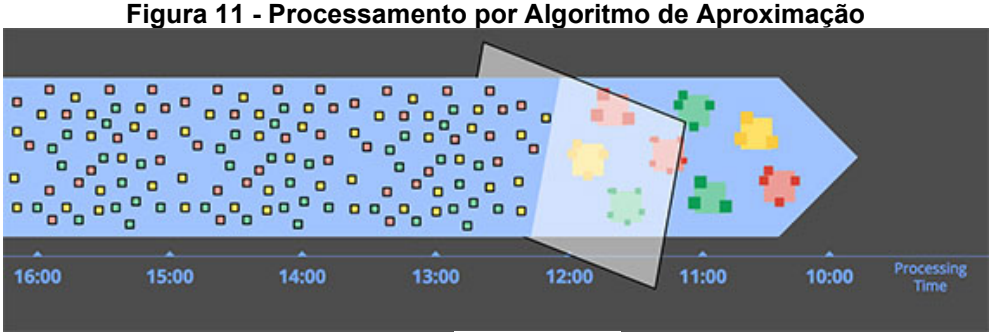


Fonte: Akidau (2015)

9.3 Processamento por algoritmo de aproximação

Em alguns casos, nos quais os parâmetros que estão sendo procurados nos dados são desconhecidos ou variam no tempo, o uso de algoritmos como Aproximação *Top*-N ou *Streaming* K-*means* permite aglomerar os dados em torno de certas tendências encontradas no fluxo. Esses algoritmos possuem um sistema de decaimento de informação (*forgetfullness*) que permite que as tendências se alterem conforme novos dados derivam do fluxo inicial.

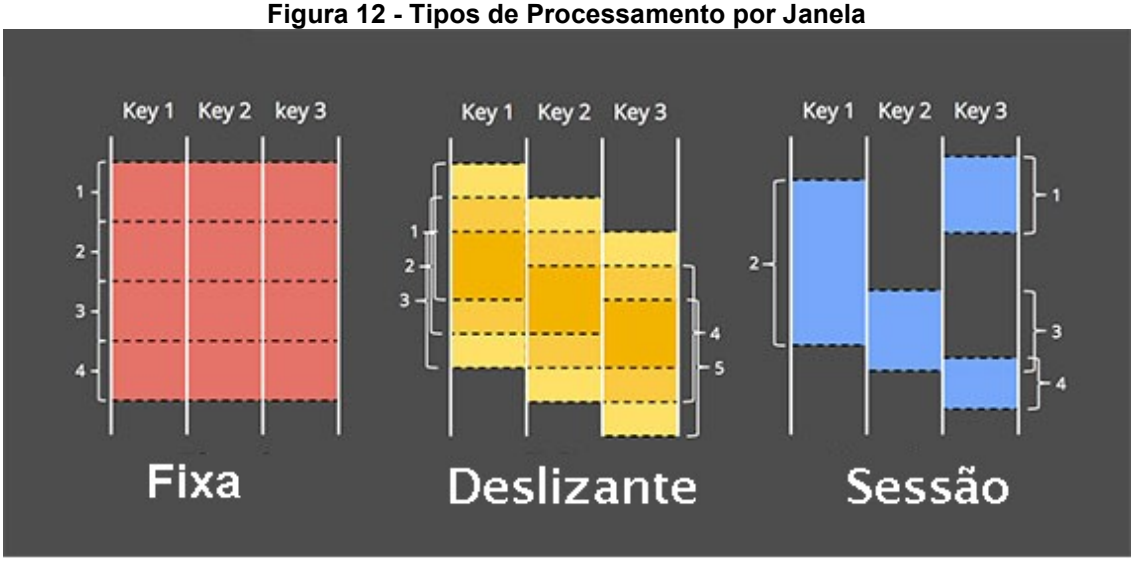
Figura 11 - Processamento por Algoritmo de Aproximação



Fonte: Akidau (2015)

Há, ainda, diversos tipos de Processamento por Janela, conforme figura 12:

Figura 12 - Tipos de Processamento por Janela



Fonte: Akidau (2015)

No processamento por janela existe um período de processamento (sinalizado na Figura 12 pelos números 1, 2, 3 e 4) onde o *Data Stream* é dividido entre diversos núcleos de processamento dentro do *cluster*, identificados por uma

chave (na Figura 12, sinalizados como *Key* 1, *Key* 2 e *Key* 3). A forma como acontece o Gatilho para iniciar o processamento e o modo como a Janela é definida segue um dos três modelos a seguir.

a) **Janela de Tempo Fixa (Micro-*Batch*):** o *Data Stream* é dividido em segmentos (1, 2, 3, 4, etc.) com tamanhos temporais fixos e distribuídos no *cluster* para processamento com uma chave de identificação (*Key* 1, *Key* 2, *Key* 3). Os dados são processados e reagrupados de acordo com sua chave pela arquitetura. Este processo permite escolher a latência do sistema para saída da informação, mas incorre no erro de não ser capaz de detectar e lidar com dados recebidos fora da sua janela de tempo.

b) **Janelas Deslizantes:** as janelas se caracterizam por um Registro de Tempo Inicial e um Final, esses registros de tempo são atualizados em intervalos regulares e os dados com seu Registro de Criação fora do valor do intervalo são descartados. As Janelas de Tempo podem se sobrepor umas às outras para garantir o processamento de dados que acabem chegando atrasados. Este sistema é aplicado quando os dados possuem valores que interagem entre si, permitindo que os dados sejam alimentados continuamente e processados dentro da janela, excluindo os dados já expirados. O uso de Registros de Persistência permite que os dados processados atualizem o registro em tempo real e que o processamento inclua os Registros de Persistência como um histórico do comportamento.

c) **Janela por Sessão:** o sistema acumula dados baseado em um parâmetro de quanto tempo será necessário para processamento e permite absorver picos de atividade, manter uma latência de atualização constante das informações e manter um sistema com a carga de trabalho distribuída de forma uniforme dentro do *cluster*.

10 ESTUDO DE CASO: PLATAFORMA *BIG DATA* DA UBER, MAIS DE 100 *PETABYTES* COM LATÊNCIA DE MINUTOS

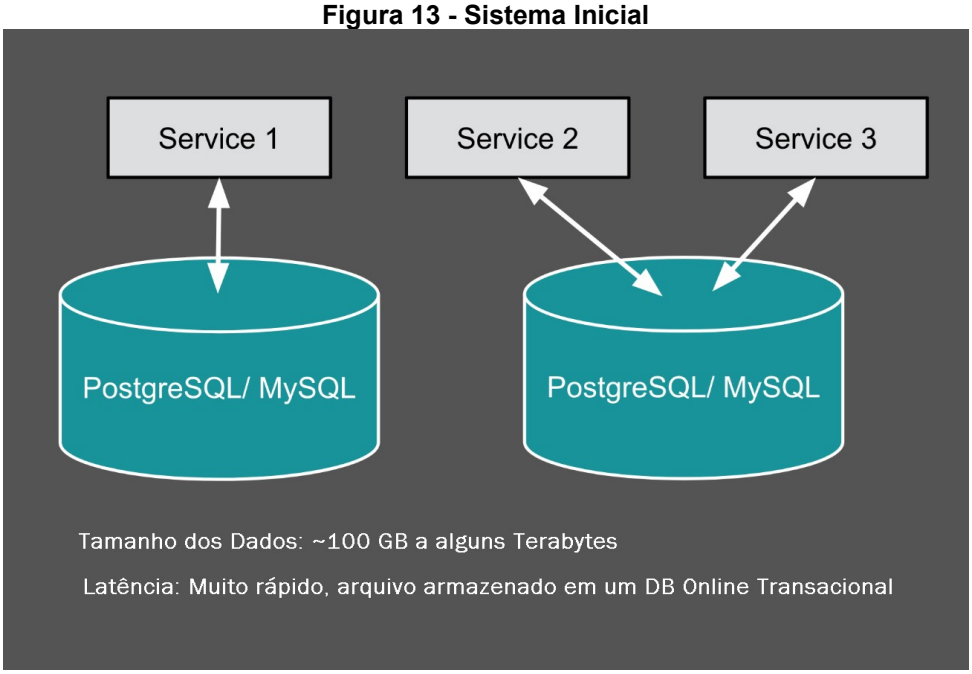
A Uber tem o objetivo de entregar transportes rápidos e seguros em diversos mercados globais. Para alcançar este objetivo, decisões direcionadas a dados (*data-driven decisions*) devem ser tomadas usando dados em tempo real com auxílio de ferramentas de previsão.

Cerca de 100 *petabytes* de dados analíticos são processados nas plataformas *Big Data* usando *Hadoop*. O processo para criar um sistema confiável de baixa latência passou por três gerações, onde problemas foram identificados e novas soluções foram propostas.

10.1 Problema inicial

Em 2014, os dados eram armazenados em servidor de transação *online* (MySQL e PostgreSQL). Para acessar os dados, cada *database* ou tabela deveria ser acessada individualmente e códigos eram escritos para realizar a combinação de diferentes bases. Não havia acesso aos dados globais e cada região tinha sua *database*, conforme Figura 1.

Figura 13 - Sistema Inicial



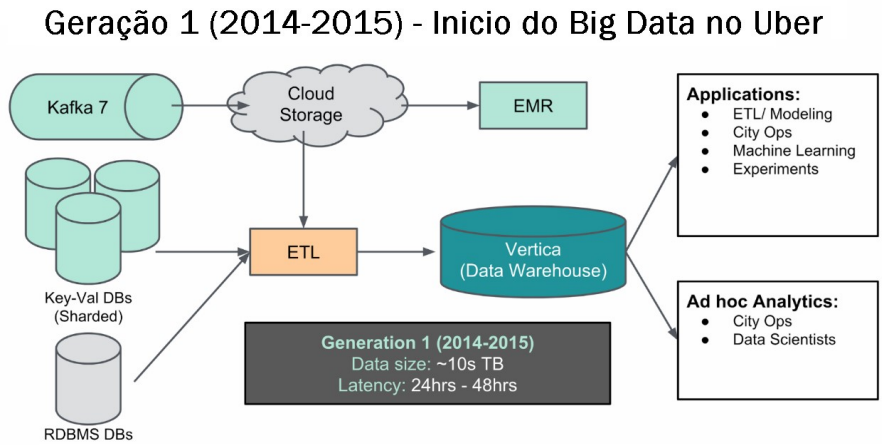
Fonte: Shiftehfar (2018)

Com o crescimento exponencial, os dados cresceram e a análise dos dados foi priorizada. A capacidade de analisar os dados em um local centralizado era necessária para tornar a empresa com uma cultura direcionada aos dados.

10.2 Solução

A primeira geração se apoiou em uma construção de um *Data Warehouse* para centralizar os dados. A arquitetura se baseou em uma *engine* ETL que copiava dados de fontes diferentes (armazenagem em nuvem, *databases*, *logs* de serviço, etc.) usando uma interface baseada em SQL para processar queries.

Figura 14 - Primeira Geração do Uber



Fonte: Shiftehfar (2018)

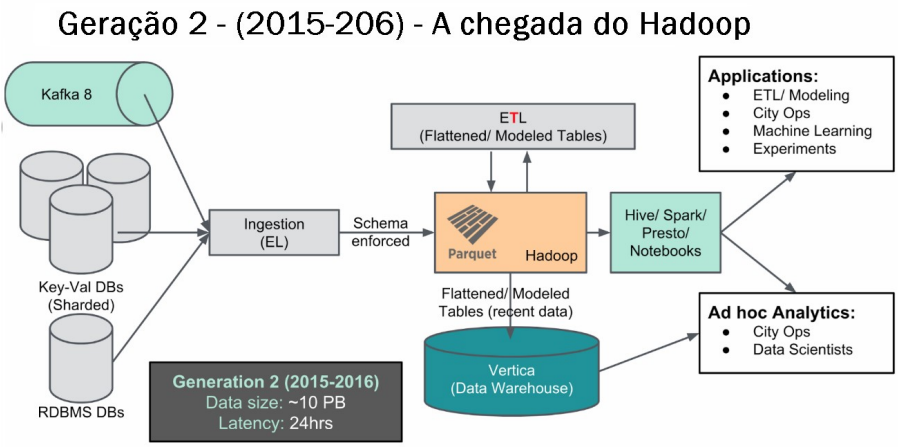
10.3 Geração 1 – *Data Warehouse*

Com a *Data Warehouse*, usuários do sistema tiveram acesso de forma centralizada a dados globais e o uso de Linguagem *Standart Query Language* (SQL) como padrão, o que abstraiu o uso da plataforma sem precisar de grande conhecimento técnico. Com a ferramenta, times passaram a desenvolver tecnologias novas e os dados cresceram para dezenas de *terabytes* rapidamente, evidenciando as limitações da nova arquitetura.

Como os dados entravam sem padronização, a consistência dos mesmos se tornou uma preocupação, pois estes estavam sendo processados de diversas formas durante a ingestão, estressando o sistema e gerando cópias redundantes e o

consequente aumento no custo de armazenamento, obrigando a deleção de dados antigos. A solução não era escalável e segura.

Figura 15 - Segunda Geração do Uber



Fonte: Shiftehfar (2018)

10.4 A Segunda Geração – Introdução do *Hadoop*

Usando o ecossistema *Hadoop*, uma reengenharia foi feita, na qual todos os dados são processados uma única vez com o uso de *Apache Spark* para acessar os dados usando queries no formato SQL e Não-SQL e a ferramenta *Apache Hive* para grandes queries.Foram introduzidas ferramentas com finalidades diferentes para o usuário escolher a melhor forma de processar suas queries.

Para manter a plataforma escalável, todo modelamento de dado e transformação *acontece uma única vez no Hadoop para evitar reprocessamentos e o uso do Apache* Parquet como padrão resulta em melhor compressão de dados. Apenas tabelas críticas eram transferidas para a *Data Warehouse*, diminuindo significantemente o custo. O uso da nova arquitetura tornou o sistema escalável, com dados padronizados e usando um sistema central com processamento em lotes unificado no *Hadoop*.

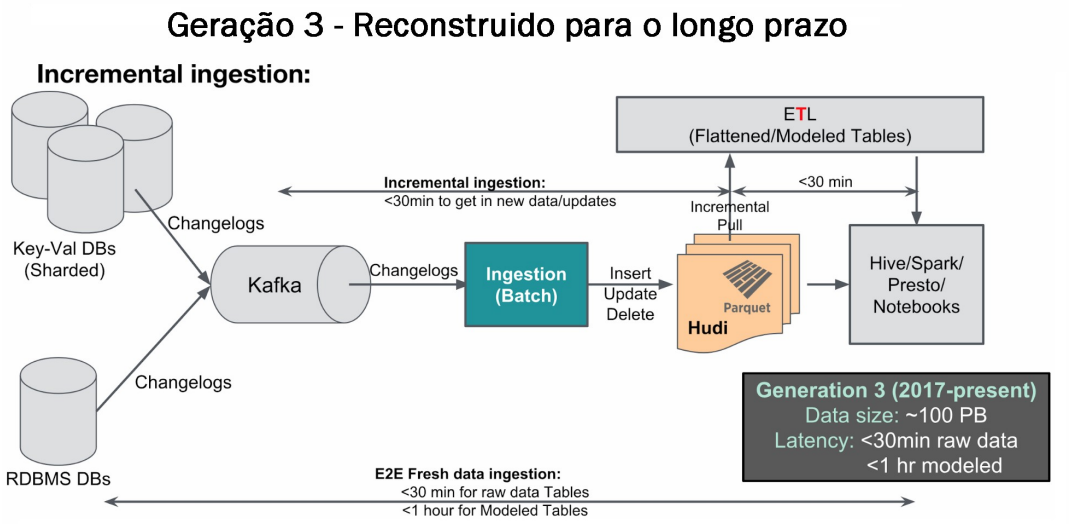
10.5 Novos desafios

Com a expansão do mercado, a plataforma alcançou o volume de dezenas de *petabytes* de dados acrescentados diariamente e os processadores virtuais do *cluster* cresceram de 10.000 para 100.000. Novas limitações surgiram, a quantidade

de pequenos arquivos armazenados no HDFS criou grande pressão sobre o sistema e o serviço de processamento demorava 24 horas para processar todos os arquivos.

O sistema ETL precisava recriar a tabela sempre que executava os *Jobs* e a atualização ocorria em todo o *dataset*, mesmo quando a mudanças era em uma única tabela. Mesmo que apenas 100 *gigabytes* de dados fossem atualizados em cada tabela, era preciso processar 100 *terabytes* para atualizar todo o *dataset* com o uso de 1000 executores *Spark* e latência de vinte horas.

Figura 16 - Terceira Geração do Uber



Fonte: Shiftehfar (2018)

10.6 Geração 3

Em 2017, a plataforma estava próxima de seu limite de escalonamento com 10.000 *Jobs* de *Spark*, 100.000 processadores virtuais e 100.000 queries por dia. Abaixo são listadas algumas limitações encontradas:

a) Escalabilidade do *Hadoop Distributed File Sys*tem (HDFS)

- Problema: A escalabilidade do HDFS depende da arquitetura de *NameNode* que é afetada pela manipulação de arquivos pequenos, geralmente quando os dados passam de 100 *petabytes*.

- Solução: Uso de um novo serviço de gerenciamento de *NameNode* chamado NHDFS *NameNode Federation* e separação de alguns dados para *clusters* diferentes.

b) Dados mais rápidos no *Hadoop*

- Problema: a Uber trabalha em tempo real e a latência de 24 horas não condiz com a realidade do serviço. Para realizar a atualização do sistema, é preciso atualizar toda a tabela em um processo por lote que leva 24 horas. O acesso aos dados é feito por meio de horários e datas, e como os dados são atualizados diretamente nos campos, o único meio de verificar se foi atualizado é verificando toda a tabela.
- Solução: Para diminuir a latência do sistema, foi introduzido um sistema de atributos baseado em Registro de Tempo (*TimeStamp*), Chave, Versão, Informação de *Data Center* e Origem do Dado onde um Ponto de Checagem com Registro de Tempo é atualizado por micro-lotes processados em janelas de tempo de 10 a 15 minutos.

Os dados da tabela são atualizados por incrementos ao invés de um processamento em lote único, o que diminui o tempo de processamento e garante uma latência de apenas 30 minutos (considerando um ciclo de 15 minutos de processamento e um ciclo de espera para gerenciar falhas no processo).

Com esses progressos, uma busca não precisa esperar um processamento completo, mas pode-se buscar um dado da tabela e realizar seus incrementos por meio das atualizações sinalizadas no Registro de Tempo, criando uma camada de consulta rápida.

- Resultado

O uso de um processamento em micro-lotes por janela de tempo fixa com atualizações baseadas em incrementos listados em um *log* de Registro de Tempo permite realizar consultas rápidas em apenas 30 minutos, garantir escalabilidade, redução de custo de armazenagem ao gravar somente dados já processados e qualidade de dados ao centralizar em um único serviço de processamento.

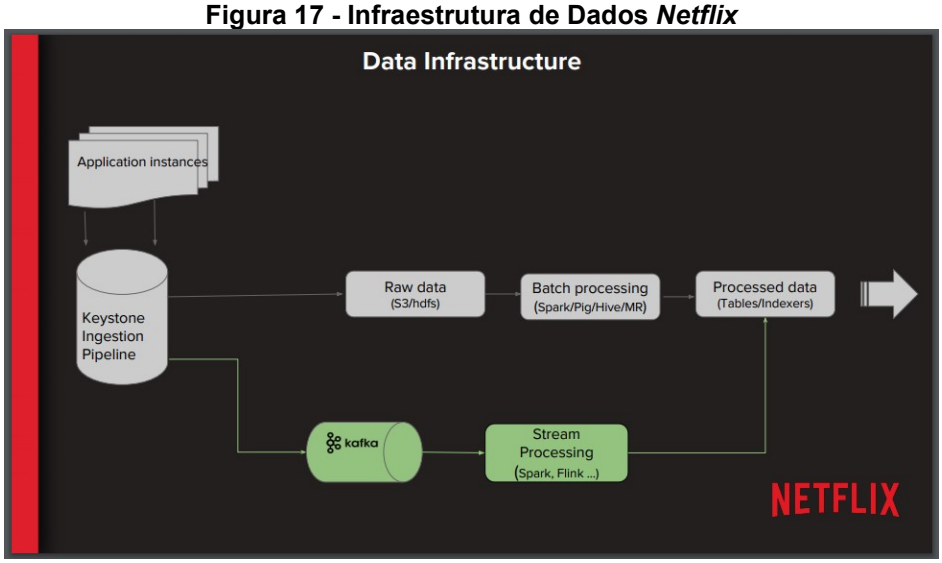
11 ESTUDO DE CASO *NETFLIX*: MIGRAÇÃO DE SISTEMA *BATCH* ETL PARA PROCESSAMENTO POR *STREAMING* USANDO *KAFKA* COM *FLINK*

A *Netflix* é uma provedora de serviços de *streaming* global que abrange 190 países e possui uma base de 203.67 milhões de usuários. Seu objetivo principal é entreter os usuários com conteúdo personalizado em qualquer lugar a qualquer tempo. Para isso, são processados 450 bilhões de eventos únicos diariamente de 100 milhões de membros ativos em 190 países.

O sistema da *Netflix* usa uma arquitetura baseada em micro serviços e comunicação remota por *procedure call* e mensageira. O sistema de produção possui um *cluster Apache Kafka* com mais de 700 tópicos que processa a mensageria e múltiplos pipelines de informações. As aplicações de micro serviço emitem eventos com informações direcionadas a sistema e usuário que são coletadas pelo *Netflix Keystone data pipeline*, um sistema de processamento de eventos em *streaming* em tempo real escalável.

Os dados em lote tradicionais são armazenados usando o *Hadoop Distributed File System* e processados com *Apache Spark*, *Pig*, *Hive* e *Hadoop*, enquanto o fluxo de dados é processado pelo *Apache Kafka* com *Apache Flink* e *Spark Streaming*.

Figura 17 - Infraestrutura de Dados *Netflix*



Fonte: Arora (2017)

O objetivo principal da transição foi permitir o uso de algoritmos de *Machine Learning* e redução de custo em armazenagem de dados (pois os dados não precisam ser guardados de forma não tratada), monitoramento em tempo real de valores chaves e integração com outros sistemas.

11.1 Desafios principais

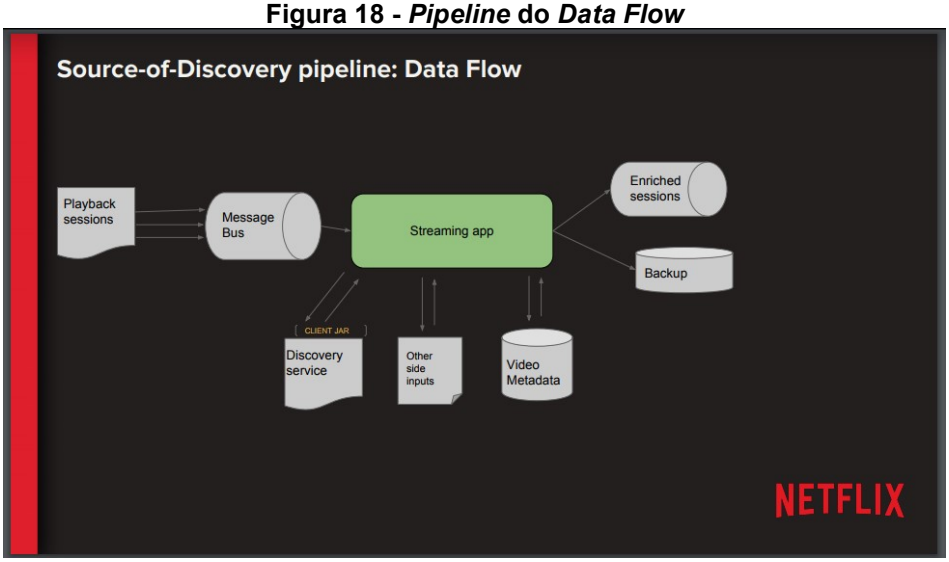
O principal desafio encontrado foi a escolha de uma *engine* apropriada, dependendo se os dados processados seriam um *Data Stream* baseado em eventos ou micro-lotes, além da redução do tempo de latência de 24 horas entre a geração de um evento e seu processamento.
A ferramenta escolhida precisa suportar uma entrada de 100 milhões de eventos por dia, se comunicar com micro serviços por meio de clientes *THICK* (RPC-*style*) ser integrável ao ecossistema da plataforma *Netflix*, possuir um gerenciador centralizado de *logs* e alertas.

11.2 Desenvolvimento

Como a empresa já utilizava um sistema de processamento em lotes, o uso de micro-lotes seria a solução mais próxima do sistema atual. A escolha do uso de micro-lotes exigia uma adoção de método cujo tamanho fosse ideal para os lotes e um estudo conduzido chegou à conclusão que o uso de janelas baseadas em sessões (*session-based windowing*) de dados de evento era o mais econômico.

Todas as *engines* de *Data Stream* suportam com variado grau de sucesso esta abordagem e a escolhida foi o *Apache Flink* pelo seu excelente suporte a customização de janelas. Após analisar as *engines* disponíveis, foi determinado o uso da arquitetura a seguir:

Figura 18 - *Pipeline* do *Data Flow*



Fonte: Arora (2017)

- *Apache Kafka* como terminal de mensageria

- *Apache Hive* para sumarização de dados, buscas e análise usando uma interface estilo SQL.

- *Amazon* S3 para armazenar os dados usando HDFS

- *Netflix* OSS para integração com o ecossistema *Netflix*.

- *Apache Mesos* para execução e programação de *Jobs*

- *Spinnaker* para entregas contínuas.

A implantação encontrou dificuldades com alguns pontos, quais sejam:

a) Ao buscar dados em tempo real

- Problema: A migração do *Job* requeria acesso ao histórico de visualizações completo do usuário em cada evento de inicialização de *playback*.

- Resolução: Devido à natureza de tempo real dos dados, uma chamada simples tipo RPC era requerida para cada evento processado.

Como *Apache Flink* é escrito em JAVA API e o *Netflix* OSS também é escrito em Java, a compatibilidade entre *libraries* das aplicações enfrentou conflitos (conhecido como "*JAR hell*").

b) Recuperação de dados

- Problema: Recuperação de dados devido a falhas é difícil, pois ao contrário do processamento em lote, onde os dados são armazenados, os dados RAW podem ser descartados após o processamento.

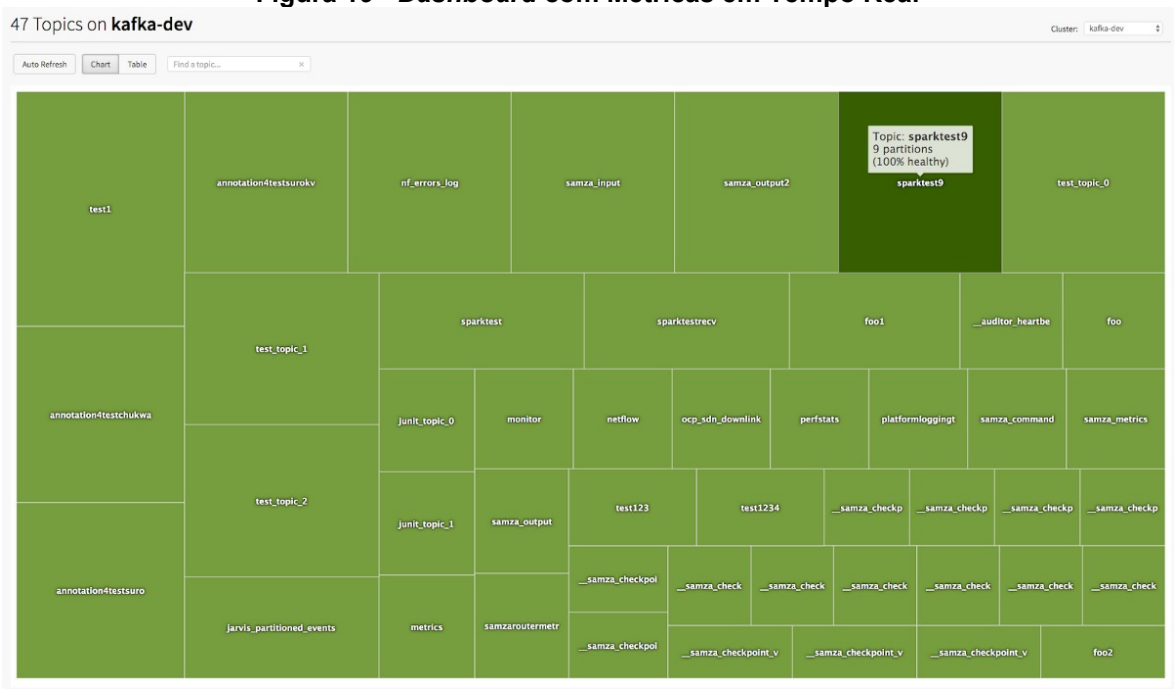
- Resolução: Os dados são estocados de forma RAW no HDFS por um tempo limitado (um a dois dias) como *backup*.

c) Eventos fora de ordem

- Problema: Caso ocorram falhas no *pipeline*, os dados antigos se misturam aos dados em tempo real.

- Resolução: O uso de uma Janela de tempo com inclusão de reprocessamento dos dados garante que os eventos são processados no contexto de tempo de evento correto. Isso reduziu o tempo para detecção de problemas de 8 minutos para 1 minuto.

Figura 19 - *Dashboard* com Métricas em Tempo Real



Fonte: Arora (2017)

11.3 Resultados

Utilizando a nova arquitetura escalável e o processamento por micro-lotes, a *Netflix* passou dos números iniciais para o processamento de cerca de 700 bilhões de mensagens em um dia, usando um sistema que permite visualizar os metadados em tempo real com monitoramento de erros no serviço de cerca de 1 minuto de latência.

12 ANÁLISE DAS VANTAGENS E DESVANTAGENS NO SISTEMA *DATA STREAM*

Em um mundo no qual dados são gerados em tempo real, o processamento *Data Stream* é uma ferramenta poderosa para empresas, governos e pessoas. Pode ser encontrado em:

a) Em sistemas onde os dados são maiores que a capacidade de armazenamento, o *Data Stream* é capaz de processar e extrair o valor, precisando de espaço de armazenamento menor e com economia. Exemplo: No processamento de mensagens de redes sociais, o interesse é filtrar os dados por palavras especificas no fluxo de informações, descartando os dados que não contiverem as palavras-chaves.
b) Para informações que possuem um valor atrelado ao tempo, o processamento *Data Stream* entrega valores com baixa latência. Exemplo: Monitoramento de pacientes críticos ou de Servidores, onde o estado atual tem importância crítica e precisa ter latência de segundos.
c) Em Comunicação Não-Humana, o uso de Inteligência Artificial para analisar os *Data Streams* gerados por sensores, robôs, dispositivos IoT e máquinas permite reconhecer padrões e comportamentos que são dependentes do tempo de geração do evento e o treino de *Machine Learning* ao alimentar o *dataset* de forma contínua, o que não é possível se usar um banco de dados estático.

Apesar dos grandes benefícios, a tecnologia está em processo de amadurecimento e não possui consensos relacionados a padronizações, metodologias e desempenho para sua aplicação. O campo de *Data Stream* possui grande potencial para evoluir ao embarcar novas tecnologias e algoritmos, revolucionando diversos ramos ao fornecer dados com baixa latência, grande aproximação da realidade e suporte a ferramentas poderosas, como a Inteligência Artificial. Os setores médico, logístico e *marketing* já possuem aplicações teorizadas mostrando o grande potencial da tecnologia.

13 CONCLUSÃO

Com o crescimento de tecnologias como *Internet* das Coisas e Comunicação Não-Humana, a demanda por informações em tempo real se torna natural. Para muitas aplicações, o alinhamento de baixa latência, pouca memória exigida e capacidade de processamento permitem a aplicação e desenvolvimento de novos produtos e soluções, mas é preciso o conhecimento das limitações atuais do modelo, bem como seus desafios.

O conhecimento adequado da arquitetura, ferramentas, funcionamento e limitações são fatores para seleção da melhor solução para a necessidade.

REFERÊNCIAS

AKIDAU, Tyler et al. The Dataflow Model: a practical approach to balancing correctness, latency, and cost in massivescale, unbounded, outofor. **Proceedings Of The Vldb Endowment**, Kohala Coast, Hawaii, v. 8, n. 12, p. 1792-1803, ago. 2015.

AKIDAU, Tyler. **Streaming 101**: a high-level tour of modern data-processing concepts. A high-level tour of modern data-processing concepts. O'Relly. [s.l.], 2015. Disponível em: https://www.oreilly.com/radar/the-world-beyond-batch-streaming-101/. Acesso em: 10 abr. 2021.

ARORA, Shriya. **Streaming for Personalization Datasets at Netflix**. INFOQ, [s.l.], 2017. Disponível em: https://www.infoq.com/presentations/netflix-personalization-datasets-streaming/. Acesso em: 10 abr. 2021.

BABCOCK, Brian et al. **Models and Issues in Data Stream Systems**. Stanford University. [s.l.], 2002. Disponível em: https://www.researchgate.net/publication/221559970_Models_and_Issues_in_Data_Stream_Systems/link/0c9. Acesso em: 23 maio 2021.

CERN (Geneva). **Computing**. CERN Accelerating Science. Geneva, 2017. Disponível em: https://home.cern/science/computing. Acesso em: 13 abr. 2021.

CRUZ, Frank da. **IBM Tabulators and Accounting Machines**. Columbia University Computing History. [s.l], 2001. Disponível em: http://www.columbia.edu/cu/computinghistory/tabulator.html. Acesso em: 10 abr. 2021.

GAMA, João; GABER, Mohamed Medhat. **Learning from Data Streams**: processing techniques in sensor networks. Lituânia: Vtex Ltda, 2007. 244 p. Disponível em: https://www.academia.edu/3151656/Learning_from_data_streams_processing_techniques_in_sensor_networks. Acesso em: 20 abr. 2021.

GAMA, João; RODRIGUES, Pedro Pereira. **Data Stream Processing**. Manhattan: Springer Verlag, 2007. 14 p. Disponível em: https://www.researchgate.net/publication/236007656_Data_Stream_Processing. Acesso em: 13 abr. 2021.

GUHA, Sudipto et al. Clustering data streams: theory and practice. Theory and practice**. IEEE Transactions on Knowledge and Data Engineering,** v. 15, n. 3, p. 515-528, May-June 2003.

International Telecommunication Union (ITU). **Series y: global information infrastructure, internet protocol aspects and next-generation networks**. ITU – International Telecomunication Union. [s.l.], 2012. Disponível em: https://www.itu.int/rec/T-REC-Y.2060-201206-I. Acesso em: 23 maio 2021.

HOLST, Arne. **Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025 (in zettabytes)**. Statista. [s.l.], 2020. Disponível em: https://www.statista.com/statistics/871513/w. Acesso em: 10 junho 2021.

JOHNSON, Joseph. **Global digital population as of January 2021** (in billions). Statista. [s.l.], 2021. Disponível em: https://www.statista.com/statistics/617136/digital-population-worldwide/. Acesso em: 23 maio 2021.

KITCHIN, Rob; MCARDLE, Gavin. What makes Big Data, Big Data?**:** exploring the ontological characteristics of 26 datasets. **Big Data & Society,** June 2016.

LAPUSAN, Tudor. **Hadoop MapReduce deep diving and tuning**. Today Software Magazine. [s.l.], 2019. Disponível em: https://www.todaysoftmag.com/article/1358/hadoop-mapreduce-deep-diving-and-tuning. Acesso em: 10 abr. 2021.

LUETH, Knud Lasse. **Why the Internet of Things is called Internet of Things**: definition, history, disambiguation. Definition, history, disambiguation. Iot Analytics. [s.l.], 2014. Disponível em: https://iot-analytics.com/internet-of-things-definition/#:~:text=The%20birth%20of%20IoT&text=But%20the%20actual%20term%20. Acesso em: 23 maio 2021.

MAINFRAME Concepts. **z/OS Basic Skills Information Center**. Poughkeepsie: Ibm Corporation, 2008.

ON, Bill. **Corporate Information Factory (CIF) Overview**. Web Archive. [s.l.], 2007. Disponível em: https://web.archive.org/web/20160623162925/http://www.inmoncif.com/library/cif/. Acesso em: 10 abr. 2021.

SHIFTEHFAR, Reza. **Uber's Big Data Platform**: 100+ petabytes with minute latency. Uber Engineering. [s.l.], 2018. Disponível em: https://eng.uber.com/uber-big-data-platform/. Acesso em: 10 abr. 2021.