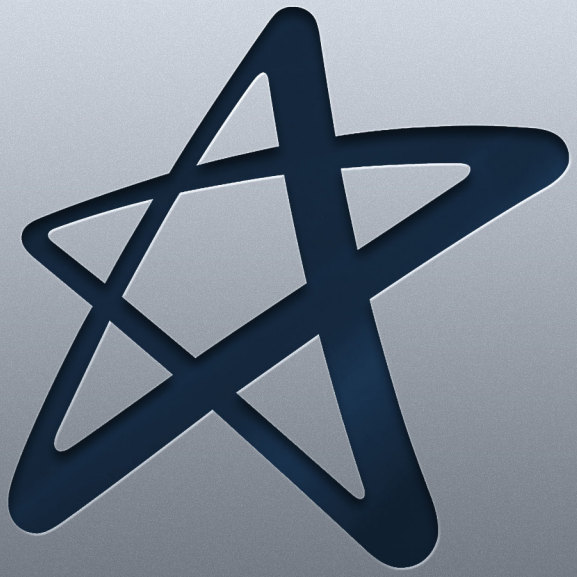
Big Data



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico

Big e *IoT*, Projeto *Hadoop* e Subprojetos

Responsável pelo Conteúdo:
Prof. Dr. Alberto Messias

Revisão Textual:
Prof.ª Dr.ª Selma Aparecida Cesarin

UNIDADE *Big* e *IoT*, Projeto *Hadoop* e Subprojetos

- Conceitos Iniciais Sobre Internet das Coisas;
- Arquitetura de IoT e Big Data;
- Introdução ao Projeto Hadoop.

OBJETIVO DE APRENDIZADO

- Compreender a relação entre a Internet das Coisas e Big Data, bem como o que é Internet das Coisas, o que são coisas e como será com IoT;
- Exemplificar aplicações para IoT e Big Data e ilustrar uma arquitetura de IoT com processamento de dados com Big Data;
- Permitir ao estudante compreender as definições iniciais sobre o projeto Hadoop, suas características e conhecer os principais projetos agregados.

Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu "momento do estudo";
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e *sites* para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Conceitos Iniciais Sobre *Internet* das Coisas

Weiser (1999) menciona que as Tecnologias mais profundas são aquelas que desaparecem.

Lyytinen e Yoo (2002) discutem que o próximo passo da computação é tornar o computador embutido nas ações naturais e nas interações humanas.

A essência da computação pervasiva reside na criação de ambientes saturados de computação e capacidade de comunicação que se integrem com a vida humana (SATYANARAYANAN, 2001).

Em abordagens preliminares, observa-se que deverão existir centenas de dispositivos por pessoa, por ambientes, de todas as escalas, conectados às Redes sem fio, tornando a computação onipresente, além da evolução das interfaces com os seres humanos que deverão ser mais transparentes e simples (WEISER, 1999).

Essa revolução não influenciará somente na quantidade de informações, mas na qualidade dela. Inúmeros e pequenos processadores embutidos em diversos objetos estarão integrados no dia a dia.

Weiser (1999) dizia que a Tecnologia é somente um meio para o propósito, ou seja, a Tecnologia é uma ferramenta. Nesse momento, definia-se o termo computação ubíqua, numa forma mais acadêmica e idealista. O termo computação pervasiva aparece como uma Tecnologia para o processamento onipresente e foi definido pela indústria (MATTERN; ZURICH, 2005), embora esses termos podem ser encarados como sinônimos.

A *IoT* é definida como uma infraestrutura de Rede Global, que interconecta fisicamente e virtualmente objetos, com o objetivo de explorar dados capturados e suas capacidades de comunicação. Essa infraestrutura inclui e envolve a *Internet* e as redes de comunicação, ela necessita de identificação única de objetos, sensores e capacidade de conexão, como base para o desenvolvimento independente de serviços e aplicações. Ela é caracterizada pelo alto grau de captura autônoma de dados, transferência de eventos de rede, conectividade e interoperabilidade (CASAGRAS, 2009).

A Figura 1 ilustra a interação existente em qualquer implementação para IoT, que incluía interação entre as Redes de Comunicação, o mundo físico e p virtual, representado pela *Internet* (CASAGRAS, 2009).

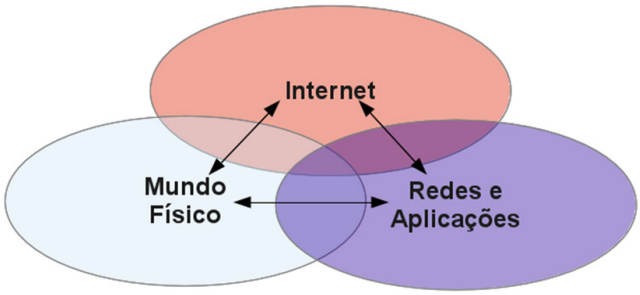


Figura 1 – Interação entre as Redes de Comunicação, o mundo físico e o virtual
Fonte: Adaptado de Casagras, 2009

O que são Coisas no Contexto de *IoT*?

A aplicação do termo Coisas inclui substâncias e produtos que são bases da sobrevivência humana e que existem em grandes quantidades e de diferentes tipos.

Vai de encontro à definição de computação pervasiva, exposta anteriormente.

Ecossistema de *IoT* Proposto pelo ITU-T

De acordo com ITU-T (2005), o ecossistema de *IoT* é composto por uma variedade de atores de negócio. Cada um desses atores possui um papel, os papéis são ilustrados pela Figura 2.

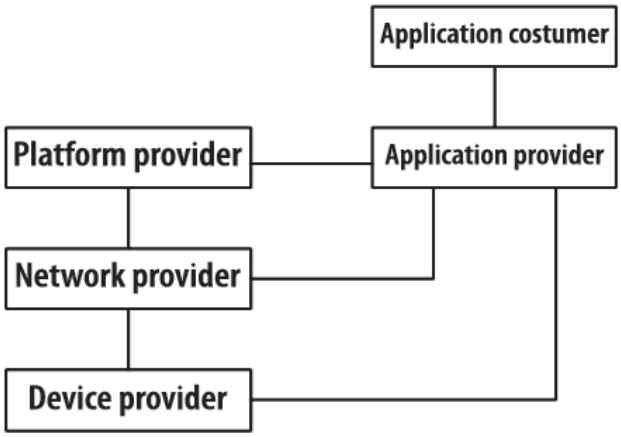


Figura 2 – Modelo de Ecossistema para *IoT*, proposto pelo ITU-T
Fonte: ITU-T, 2005

A Figura 2 ilustra os papéis de negócios presentes no Ecossistema de *IoT* e o relacionamento entre eles. Segue uma pequena descrição de cada um dos papéis.

- ***Device provider:*** o provedor de dispositivo é responsável por fornecer os dados brutos ou o conteúdo para o provedor de Rede e para o provedor de aplicação, de acordo com a lógica de negócio;
- ***Network provider:*** o provedor de Rede possui o papel central no Ecossistema de *IoT*. Ele possui as seguintes funções principais:
  - » Acesso e integração de recursos providos por outros provedores;
  - » Suporte e controle das capacidades de infraestrutura de *IoT*;
  - » Oferecer capacidades de *IoT*, incluindo capacidades de Rede e exposição de recursos para outros provedores;
- ***Platform provider:*** o provedor de plataforma provê capacidades de integração e interfaces abertas. Diferentes plataformas podem prover diferentes capacidades para provedores de aplicação. Capacidades de plataforma incluem, tipicamente, capacidades de integração, como, por exemplo, armazenamento de dados, processamento de dados ou gerenciamento de dispositivos e suporte para diferentes tipos de aplicações de *IoT*;

- **Application provider:** o provedor de aplicação provê e utiliza capacidades ou recursos providos por provedores de Rede, provedores de dispositivos e provedores de plataformas, de modo a prover aplicações de *IoT* para aplicações clientes;
- ***Application customer:*** a aplicação cliente é o usuário das aplicações de *IoT* providas pelos provedores de aplicação; nesse caso, uma aplicação cliente pode representar múltiplas aplicações de usuário.

Conhecer esses serviços principais será importante para fazer uma conexão com a Tecnologia de *Big Data* ou mesmo sobre como se desenvolver ou compreender uma plataforma para *IoT*.

Quais são os Tipos de Dispositivos para *IoT*

A *IoT* refere-se à próxima geração de *Internet* (BAHGA; MADISETTI, 2014), que possuirá trilhões de nós, representados por pequenos dispositivos ubíquos ou embutidos nos diversos ambientes, dotados de sensores, interconectados a servidores *web*, supercomputadores ou *clusters*. Tecnologias como as redes de sensores, a identificação única de objetos, a comunicação móvel, a localização em tempo real, a computação ubíqua, o protocolo *IPv6* integram essa nova infraestrutura de computação e comunicação.

A Figura 3 ilustra o Modelo Inclusivo para a *IoT*, adaptado de Souza (2015).

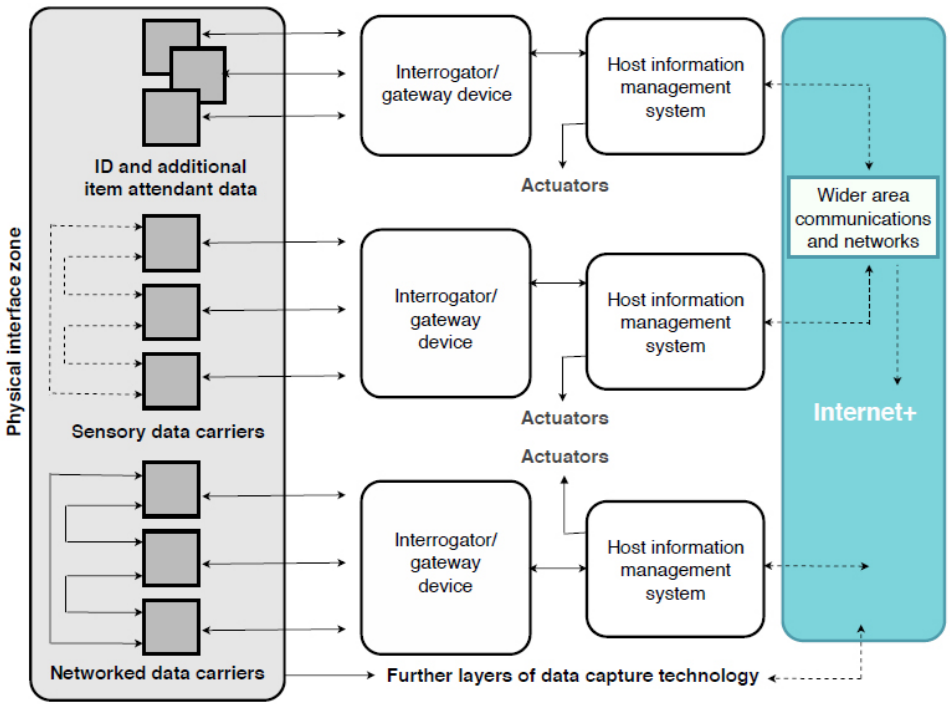


Figura 3 – Modelo Inclusivo para a *IoT*
Fonte: Adaptado de Souza, 2015

A Figura 3 representa o Processo de Captura e o Processamento de Dados em *IoT*, que ocorre da seguinte maneira: os dados gerados pelos objetos físicos são lidos pelo dispositivo interrogador, também designado como *gateway*.

Esses dados são enviados ao Sistema de Gerenciamento de Informação. A partir dele, usando a *Internet* ou outras Tecnologias de Rede, são obtidos outros dados associados aos objetos ou aos próprios dados iniciais. O resultado do processamento por uma aplicação ou serviço é uma atuação no ambiente em que estão presentes os objetos e/ou uma atuação sobre eles.

Conforme se observa em Souza (2015), são propostas três classes de dispositivos para *IoT*:

1. Objetos puramente passivos com identificação e dados fixos, são objetos presentes na *Physical Interface Zone*, na parte superior da Figura 3;
2. Objetos dotados de moderado poder computacional e percepção de contexto que, por meio de sensores, podem gerar mensagens e variar a informação associadas a eles de acordo com o tempo e lugar, representados pelos objetos presentes na *Physical Interface Zone*, mais ao centro da Figura 3;
3. Objetos que possuem conectividade em Rede, sem a intervenção humana, possibilitando a emergência de inteligência nos Sistemas de Rede, representados pelos objetos presentes na *Physical Interface Zone*, na parte inferior da Figura 3.

Como aplicações para *IoT*, sugerem-se diversas áreas, a saber:

- Casas inteligentes;
- Vida assistida;
- Aplicações médicas;
- Indústrias inteligentes;
- Segurança e controle de acesso;
- Carros inteligentes;
- Tráfego inteligente;
- Cidades inteligentes;
- Compras automatizadas;
- Logística inteligente;
- Monitoramento de vida animal;
- Cenários para estimação, alertas e recuperação de desastres naturais;
- Uso de energia elétrica, água, gás;
- Ambientes inteligentes com monitoramento e controle de gasto de energia;
- Sistemas de entretenimento inteligente;
- Sistemas para agricultura inteligente.

Note que essas áreas são as mais diretas ao se pensar em aplicações para *IoT*; porém, não se restringem a elas.

Como Será e como Desenvolver Aplicações para *IoT*

Segue um exemplo de como será a transmissão de dados com *IoT*. A *IoT* envolve uma grande quantidade de nós; sua utilização produzirá um grande volume de dados.

Como exemplo, cito um processo de Cadeia de Suprimentos num Hipermercado, na qual necessitamos de alguns dados, como a identificação dos objetos, a informação de localização e o tempo em que o objeto esteve à venda. Criando uma estrutura de dados simples, para armazenar ou enviar esses dados, serão necessários 18 *bytes* de dados de um único objeto.

Caso esse hipermercado possua em torno de 700.000 produtos, para esses dados serão gerados 12,6*GB* caso a frequência de leitura dos dados seja a cada segundo. Serão gerados 544TB a cada doze horas, um volume expressivo para a infraestrutura de Rede Local.

As informações geradas pelos diferentes dispositivos deverão ser processadas ou guardadas para um processamento posterior. Observe que serão necessários novos mecanismos eficientes e políticas inerentes à própria infraestrutura de rede para o Gerenciamento, Armazenamento e Processamento desses dados.

O modelo de Redes para tráfego de dados e informação deverá caracterizar-se como trafego de dados fim a fim; porém, possuirá vários atores, como nós de origem de dados, coletores de dados e pontos intermediários de processamento.

Os fluxos de dados poderão ser alterados nos nós finais ou intermediários ao longo do caminho. Mas, notadamente, comunicação máquina a máquina.

De acordo com a Literatura, observa-se que para suportar e dar escala às aplicações de *IoT*, é importante ter uma camada de *middleware*.

Segue um exemplo: caso o dado de um único sensor de temperatura, com baixo poder computacional, seja importante numa aplicação para dispositivos móveis e essa esteja instalada em 10 mil dispositivos, se a frequência de leitura desse dado seja a cada segundo, o sensor específico não conseguirá atender às requisições. Será necessária uma camada de *middleware* que possa dar escala à aplicação e responder todas as requisições.

A Figura 4 ilustra a arquitetura proposta.

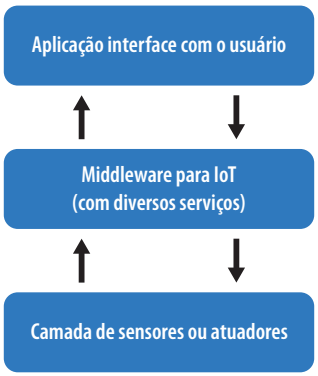


Figura 4 – Arquitetura ilustrativa para o desenvolvimento de serviços e aplicações baseadas em *IoT*

Na Figura 4, vê-se uma camada inferior que representa a camada física com sensores ou atuadores, que deve implementar ou ter um *Gateway* de *IoT* que posso fazer a interface com o *middleware*.

Na camada central, encontra-se o *middeware*, que provê alguns serviços do modelo e, na camada superior, a aplicação que também deve implementar a comunicação com o *middleware* para receber os dados ou enviar comandos para o *middleware*.

Como exemplos de *middleware* específicos para *IoT* temos o *Link Smart Middleware* (SARNOVSKY *et al*., 2008), ou a plataforma *UbiDots* (UBIDOTS, 2017), ambos disponíveis na *Web* para utilização.

Arquitetura de *IoT* e *Big Data*

Como seria uma arquitetura que suportaria esta integração?

Note que a integração entre as diversas camadas do modelo de *IoT* deverão ser altamente integradas às plataformas de *IoT*, computação em nuvem, *Big Data* e algoritmos ou plataformas para reconhecimento de padrões.

Destaco, ainda, o uso dos *middlewares* ou plataformas para integração de dispositivos na *Internet* dos grandes provedores de serviços de computação em nuvem, como, por exemplo:

- *Azure IoT Central*, da *Microsoft*, que concentra uma série de serviços para *IoT* na plataforma *Azure* https://goo.gl/2CsKBz.

Note que essas plataformas já permitem a integração entre os serviços de:

- Recepção e armazenamento do grande volume de dados;
- Processamento e tratamento dos dados;
- Aprendizagem por máquina para a análise dos dados;
- Apresentação das informações;
- Comunicação de retorno para o mundo real em *IoT*;

Um dos ecossistemas importantes para essa conexão com a Tecnologia de *Big Data* é a plataforma *Haddop*.

Introdução ao Projeto *Hadoop*

Hadoop é o termo usado para se referir a uma família de projetos relacionados, que compõe a infraestrutura para Computação Distribuída e de larga escala de Processamento, que usa o conceito de *Big Data*.

De acordo com (WHITE, 2015), *Hadoop* é a implementação para *MapReduce* e sistema de arquivos distribuído mais utilizado e conhecido.

O *MapReduce* é um modelo de Processamento de Dados distribuído e um ambiente de execução em *clusters* de larga escala. Esse modelo divide o Processamento em Mapas e o divide em fases; cada fase se baseia num par de chaves/valor usado como entrada e saída para o processo. O programador especifica duas funções, o mapa e as funções de redução, para serem usadas na implementação e na execução específica (WHITE, 2015).

A Biblioteca de *software Apache Hadoop* é um *framework* que permite o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores que utilizam modelos de programação simples.

Ele é projetado para escalar a partir de um único servidor para milhares de máquinas, cada um oferecendo computação e armazenamento local. Em vez de confiar em *hardware* para proporcionar maior disponibilidade, a biblioteca em si é concebida para detectar e tratar falhas na camada de aplicação, de modo a fornecer um serviço altamente disponível no topo de um conjunto de computadores, cada um dos quais propenso a falhas.

O projeto principal é composto por alguns componentes principais, que são:

- ***Hadoop Common***: os utilitários e bibliotecas comuns que dão suporte aos outros módulos do *Hadoop*, como, por exemplo, fornece sistemas de arquivos, abstrações do Sistema Operacional, contém os arquivos e *scripts Java* necessários para iniciar *Hadoop*;
- **Sistema de Arquivos Distribuídos *Hadoop* (*HDFS*)**: um Sistema de Arquivos Distribuídos que fornece acesso de alto débito aos dados do aplicativo;
- **Hadoop YARN**: uma estrutura para agendamento de trabalho e gerenciamento de recursos de *cluster*;
- ***Hadoop MapReduce***: um Sistema baseado em *YARN* para Processamento paralelo de grandes conjuntos de dados. Funciona como uma *API* que permite a criação de Programas que executarão a divisão e a redução das tarefas, executando sempre a estrutura de entrada e saída presente no *Hadoop*.

A Figura 5 ilustra a estrutura do *MapReduce*.

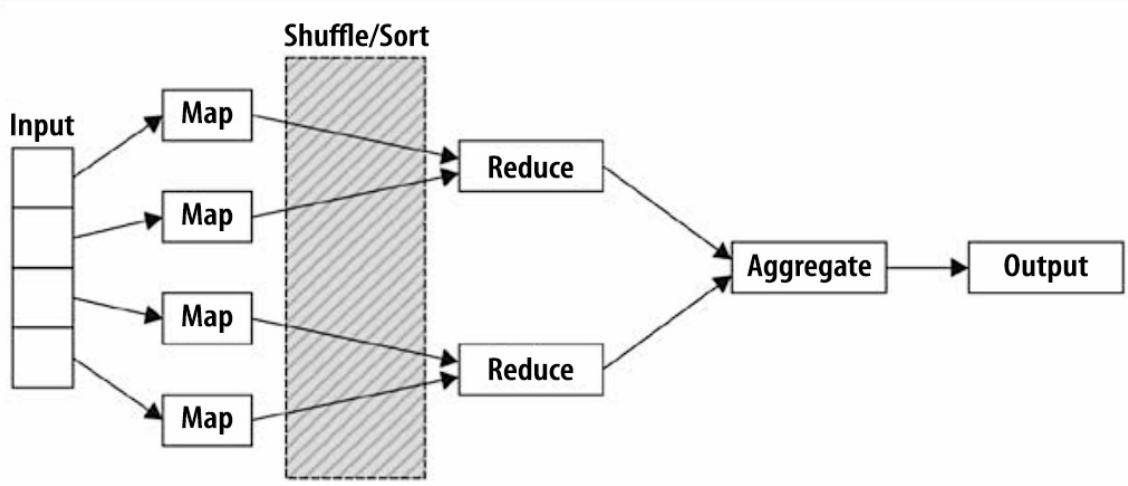


Figura 5
Fonte: Adaptado de Shenoy, 2014

Pela Figura, observa-se a divisão das tarefas e a posterior agregação dos resultados para criar uma única saída.

A Figura 6 ilustra a estrutura básica do *Hadoop* e as interações entre os principais projetos:

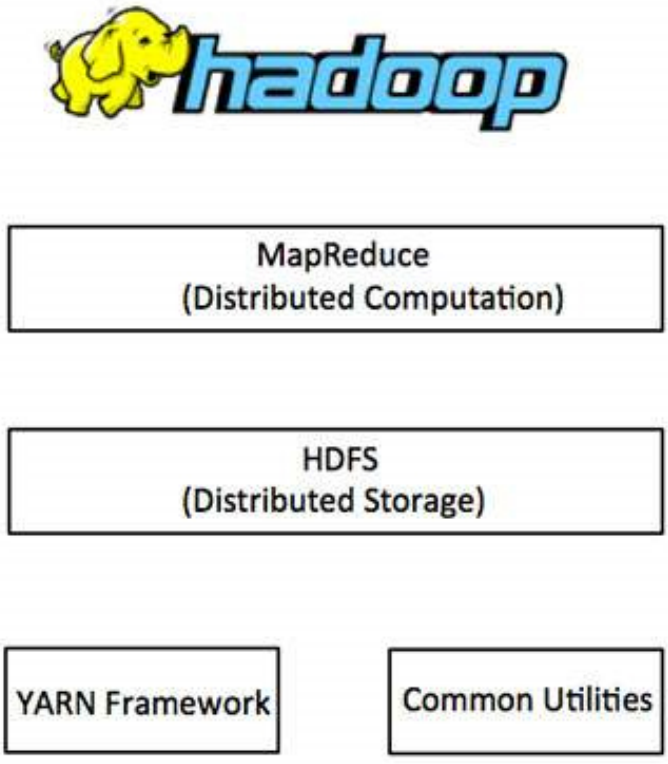


Figura 6.
Fonte:

Hadoop Distributed File System (HDFS)

O *HDFS* é um Sistema de Arquivos projetado para armazenar arquivos extremamente grandes com um padrão de fluxo de acesso, executar sob *clusters* de computadores pessoais ou plataformas de *hardware* comuns (WHITE, 2015).

O *HDFS* é um Sistema de Arquivos altamente tolerante a falhas, projetado para executar em *hardware* padrão de baixo custo. É ideal para armazenar grandes quantidades de dados; permite a conexão de nós contidos nos *clusters* por meio dos quais os arquivos de dados são distribuídos. É possível acessar e armazenar os Arquivos de Dados como um Sistema de Arquivos contínuo.

Seguem algumas características a destacar:

- **Arquivos muito grandes:** "Muito grande", nesse contexto, significa arquivos que são centenas de *megabytes*, *gigabytes*, ou *terabytes* de tamanho. Acesso a dados de *streaming*: *HDFS* é construído em torno da ideia de que o Padrão de Processamento de Dados mais eficiente é uma *write-once*, muitas vezes, ler-padrão. Um conjunto de dados é tipicamente gerado ou copiado a partir da fonte; em seguida, várias análises são realizadas em conjunto de dados ao longo do tempo. Cada análise envolve uma grande parte, se não a totalidade, do conjunto de Dados, de modo que o tempo para ler o conjunto de dados inteiro é mais importante do que a latência na leitura do primeiro registro;
- ***Hardware commodity*:** o *Hadoop* não exige *hardwares* caros altamente confiáveis para correr. Ele foi projetado para ser executado em *clusters* de *hardware commodity* (*hardware* comumente disponíveis de vários fornecedores), cuja probabilidade de falha do nó no *cluster* é elevada, pelo menos para grandes aglomerados. *HDFS* é projetado para continuar a trabalhar sem um interrupção perceptível para o usuário em face de tal falha.

Áreas em que *HDFS* não é bom:

- **Acesso de baixa latência de dados:** os aplicativos que requerem acesso de baixa latência aos dados, de dezenas de milissegundos gama, não vão funcionar bem com *HDFS*. Recorde, *HDFS* é optimizada para fornecer uma alta taxa de transferência de dados, e isso pode ser feito à custa de latência. HBase é, atualmente, a melhor escolha para o acesso de baixa latência;
- **Muitos pequenos arquivos:** Desde o *namenode*, contém os metadados do Sistema de Arquivos na memória. O limite para o número de arquivos em um Sistema de Arquivos é regido pela quantidade de memória no *namenode*. Como regra geral, cada arquivo, diretório e bloco leva cerca de 150 *bytes*. Assim, por exemplo, se você tivesse um milhão de arquivos, cada um tendo um bloco, você precisaria de pelo menos 300 *MB* de memória. Enquanto o armazenamento de milhões de arquivos é viável, bilhões está além da capacidade de *hardware* atual;
- **Vários escritores, modificações em arquivos arbitrários:** arquivos em *HDFS* poderão ser escritos por um único escritor. Gravações são sempre feitas no

final do ficheiro. Não há suporte para vários escritores ou para modificações em deslocamentos arbitrários no arquivo (eles podem ser suportados no futuro, mas eles tendem a ser relativamente ineficientes.)

O *MapReduce* e o *HDFS* possuem uma *API* para o desenvolvimento e o uso de suas funcionalidades.

Destaco a leitura presente no *link* da IBM, que traz algumas definições e *links* para outros artigos que tratam de plataforma *Hadoop* e outros projetos: https://goo.gl/SA5UDy

Sub-Projetos *Hadoop*

Seguem alguns dos principais dos sub-projetos *Hadoop*:

Apache Pig

Apache Pig é uma plataforma para análise de grandes conjuntos de dados, que consiste numa linguagem de alto nível para expressar Programas de Análise de Dados. Ele possui a infraestrutura para a avaliação desses Programas, sua estrutura é passível de paralelização que, por sua vez, permite lidar com grandes conjuntos de dados.

O usuário não precisa se preocupar com o desempenho, pois ele próprio faz a otimização.

A camada de linguagem do *pig*, atualmente, consiste em uma linguagem textual chamado "*Pig Latin*", que tem as seguintes propriedades-chave:

- A linguagem utilizada para expressar os fluxos de dados, chamado *Pig Latin*;
- O ambiente de execução para executar Programas na Linguagem *Latin Pig*. Atualmente, há dois ambientes: execução local em uma única JVM e execução distribuída em um *Cluster Hadoop*.

Segue um exemplo simples. Basta escrever o Programa para calcular o valor máximo da temperatura registrada por ano para o conjunto de dados em tempo *Pig Latin*:

```
-- max_temp.pig: Finds the maximum temperature by year
records = LOAD 'input/ncdc/micro-tab/sample.txt'
  AS (year:chararray, temperature:int, quality:int);
filtered_records = FILTER records BY temperature != 9999 AND
  (quality == 0 OR quality == 1 OR quality == 4 OR quality == 5 OR quality == 9);
grouped_records = GROUP filtered_records BY year;
max_temp = FOREACH grouped_records GENERATE group,
  MAX(filtered_records.temperature);
DUMP max_temp;
```

Hive

Um dos maiores ingredientes na plataforma de informação construídos pela equipe de Jeff no *Facebook* foi o *Hive*. Uma estrutura para armazenamento de

Dados em cima do *Hadoop*, o *Hive* cresceu a partir de uma necessidade de gerenciar e aprender com os grandes volumes de Dados que o *Facebook* estava produzindo todos os dias, a partir de sua crescente Rede Social. Depois de tentar alguns diferentes Sistemas, a equipe optou.

- *Hadoop* para armazenamento e processamento, uma vez que era rentável e conheceu suas necessidades de escalabilidade;
- *Hive* foi criado para torná-lo possível para analistas com fortes habilidades em *SQL* (mas habilidades de programação *Java* escassas) para executar consultas sobre os enormes volumes de dados que o *Facebook* armazenava no *HDFS*.

Hoje, *Hive* é um Projeto *Apache* bem sucedido, usado por muitas Organizações, com o propósito de ser uma Plataforma de Processamento de Dados escalável.

Seguem alguns exemplos de comandos em *Hive*:

```
CREATE TABLE records (year STRING, temperature INT, quality INT)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '\t';

  LOAD DATA LOCAL INPATH 'input/ncdc/micro-tab/sample.txt'
  OVERWRITE INTO TABLE records;
```

```
hive> SELECT year, MAX(temperature)
    > FROM records
    > WHERE temperature != 9999
    >   AND (quality = 0 OR quality = 1 OR quality = 4 OR quality = 5 OR quality = 9)
    > GROUP BY year;
1949  111
1950  22
```

HBase

HBase é um Banco de Dados orientado à coluna distribuída, construído em cima do *HDFS*. É a aplicação *Hadoop* para usar quando você precisa em tempo real de leitura/gravação de acesso aleatório para grandes conjuntos de dados. O *HBASE* é um Banco de Dados do tipo *NoSQL* e não relacional. Como exemplos de *NoSQL*, além do *HBASE*, temos o *MongoDB*, o *Amazon SimpleDB* e o *Oracle NoSQL*.

Sua melhor aplicabilidade é quando temos uma quantidade de Dados muito grande, aproveitando assim o potencial do *cluster Hadoop/HDFS*.

Com Bases de Dados que utilizem 5 nós ou mais, o *HBASE* realmente mostra o seu potencial. Podemos acessar os dados armazenados no *HBASE* por meio de uma *API Java*, por linha de comando (*HBASE Shell*), *Python*, *Avro*, *Rest*, *Thrift* etc.

A maneira mais rápida de acessá-lo é usando a *API Java*, que permite criar tabelas, fazer *Put*, *Get*, *Scan* e *Delete*, dentre outros comandos.

Segue um exemplo de comando em *Hbase*:

```
hbase(main):021:0> put 'test', 'row1', 'data:1', 'value1'
0 row(s) in 0.0454 seconds
hbase(main):022:0> put 'test', 'row2', 'data:2', 'value2'
0 row(s) in 0.0035 seconds
hbase(main):023:0> put 'test', 'row3', 'data:3', 'value3'
0 row(s) in 0.0090 seconds
hbase(main):024:0> scan 'test'
ROW                    COLUMN+CELL
 row1                  column=data:1, timestamp=1240148026198, value=value1
 row2                  column=data:2, timestamp=1240148040035, value=value2
 row3                  column=data:3, timestamp=1240148047497, value=value3
3 row(s) in 0.0825 seconds
```

Zookeeper

O Hadoop tem um serviço de coordenação distribuída, chamado *ZooKeeper*, que é um serviço centralizado para manter as informações de configuração, nomeando, proporcionando sincronização distribuída e prestação de serviços do grupo.

Altamente disponível, funciona num conjunto de máquinas e é projetado para ser altamente disponível. Ele evita a introdução de pontos únicos de falha em seu Sistema, para que você possa construir um aplicativo confiável. Interações *Zookeeper* apóiam os serviços participantes que não precisam saber um sobre o outro. Por exemplo, o *ZooKeeper* pode ser utilizado como um mecanismo de encontro, de modo que processos que de outra forma não sabem da existência um do outro (ou detalhes da Rede) possam descobrir e interagir uns com os outros.

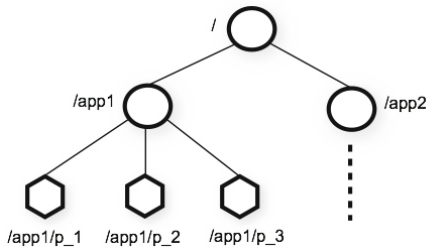


Figura 7
Fonte: apache.org

Cassandra

Cassandra é uma implementação de família de colunas *NoSQL*, que suporta o modelo de dados *Big Table* e usa aspectos de arquitetura introduzidos por *Amazon Dynamo*.

Alguns dos pontos positivos do Cassandra são:

- Altas escalabilidade e disponibilidade, sem um ponto único de falha;
- Implementação da família de colunas *NoSQL*;
- Rendimento de gravação muito alto e bom rendimento de leitura;
- Linguagem de consulta semelhante a *SQL*;
- Consistência ajustável e suporte para replicação;
- Esquema flexível;

Chukwa

É um sub-projeto dedicado ao carregamento massivo de vários arquivos de texto dentro de um *cluster Hadoop* (*ETL*). *Chukwa* se constrói sob o Sistema de Arquivos Distribuído (*HDFS*) e o marco *MapReduce* e herda a escalabilidade e robustez de *Hadoop*. Também inclui um conjunto de ferramentas flexível e potente para a visualização e a análise dos resultados.

Ambari

Ambari permite aos administradores do Sistema para disposição gerenciar e monitorar um *cluster Hadoop*, e também integrar *Hadoop* à infraestrutura corporativa existente.

Provisionando um *cluster Hadoop*:

- Não importa o tamanho do seu *cluster Hadoop*, a implantação e a manutenção dos exércitos é simplificada usando *Ambari. Ambari* inclui uma interface *Web* intuitiva, que permite que a você facilmente dispor, configurar e testar todos os serviços do *Hadoop* e componentes principais. *Ambari* também fornece a poderosa *API Ambari Blue Prints* para automatizar instalações de *cluster*, sem a intervenção do usuário.

Gerenciar um *cluster Hadoop*:

- *Ambari* fornece ferramentas para simplificar o Gerenciamento do *cluster*. A interface *Web* permite controlar o ciclo de vida de serviços e componentes do *Hadoop*, modificar configurações e gerenciar o crescimento contínuo do seu *cluster*.

Monitorar um *cluster Hadoop*:

- Ter uma visão instantânea sobre a saúde *do seu cluster. Ambari* pré-configura alertas para assistir serviços *Hadoop* e visualiza os dados operacionais de *cluster* numa interface *Web* simples.

Integrar *Hadoop* com a Empresa:

- *Ambari* oferece uma *API RESTful* que permite a integração com as ferramentas existentes, como o *Microsoft System Center Operations Manager*, *HP Operations Manager* e Teradata Ponto de Vista, para mesclar *Hadoop* com os seus processos operacionais estabelecidos.

Segue a Figura 8, que ilustra o relacionamento entre os sub-projetos agregados ao *Hadoop*.

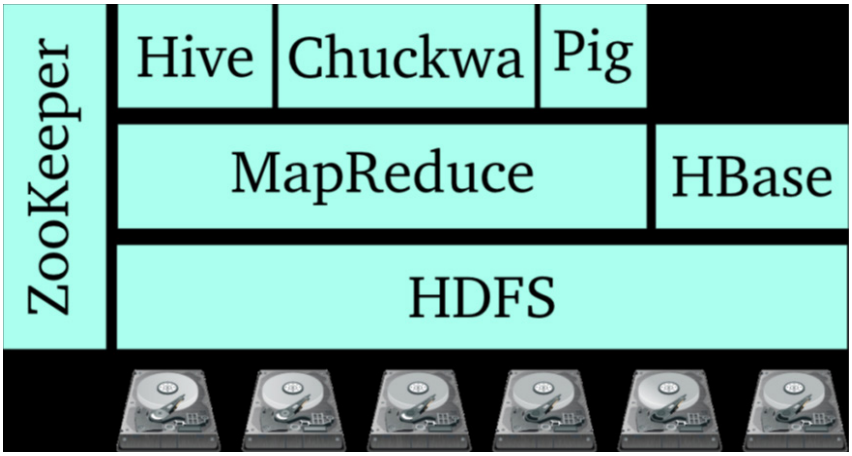


Figura 8
Fonte: Adaptado de WHITE, T. Hadoop: The Definitive Guide, 2015

A plataforma *Hadoop* possui inúmeros outros projetos que têm evoluído diariamente; alguns deles têm se destacado como projetos importante no ecossistema de *Big Data*.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Sites

Link to Leaders

Segue o *link* que elenca algumas plataformas para *IoT*.

https://goo.gl/nPHw4e

Intel *IoT* Patform

Plataforma de *IoT Intel*.

https://goo.gl/LV2xiJ

Microsofit Azure

Leitura *Microsofit* sobre o apache *Hadoop*.

https://goo.gl/z1Dwip

Hortonworks

Link para Máquina Virtual *Hadoop*, para experimentos e *download*.

https://goo.gl/5b9M2z

Apache Hadoop

Link de referência com os diversos comandos existentes para o HDFS.

https://goo.gl/BBdAzK

Leitura

O que é *IoT* (Internet das Coisas)? Futuro ou Presente?

O autor ilustra definiçoes sobre *Internet* da coisas e diversas aplicações, bem como faz uma comparação entre o presente e futuro da *IoT*.

https://goo.gl/xex996

Referências

BAHGA, A.; MADISETTI, V. ***Internet* of Things:** A Hands-On Approach. [S.l.]: Vijay Madisetti, 2014.

CASAGRAS, E. F. P. **Casagras final report**: rfid and the inclusive model for the internet ofthings. 2009.

GIACOMELLI, P. **Apache Mahout Cookbook**. [S.l.]: Packt Publishing, 2013.

HUANG, Y.; LI, G. **Descriptive models for internet of things.** INTERNATIONAL CONFERENCE ON INTELLIGENT CONTROL AND INFORMATION PROCESSING. August 2010.

ITU. **ITU: Internet of Things 2005:** executive summary. 2017. Disponível em: <https://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf>. Accesso em: 20 abr. 2017.

LYYTINEN, K.; YOO, Y. **Issues and Challenges in Ubiquitous Computing.** Communications of the ACM, v. 45, n.12. p. 63-65., 2002

MATTERN, F.; ZURICH, E. **Ubiquitous computing:** scenarios for an informatized world, communication and the media economy of the future. **Springer-Verlag**, p. 145-163, 2005.

QUINTERO, D. *et al.* **IBM Data Engine for Hadoop and Spark**. IBM RedBooks, 2016. Disponível em: <http://www.redbooks.ibm.com/redbooks/pdfs/sg248359.pdf>. Acesso em: 12 set. 2017.

SARNOVSKY, M. *et al.* **First demonstrator of hydra middleware architecture for building automation**. Hydra Project. 2008.

SATYANARAYANAN, M. **Pervasive computing:** vision and challenges. IEEE Personal Communications, 2001.

SHENOY, Aravind. **Hadoop Explained**, Packt Publishing. Mumbai. 2014. Disponível em: <https://www.packtpub.com/packt/free-ebook/hadoop-explained>. Acesso em: 12 set. 2017.

SOUZA, Alberto Messias da Costa. **Uma nova arquitetura para *Internet* das Coisas com análise e reconhecimento de padrões e processamento com *Big Data***. 2015. Tese (Doutorado em Sistemas Eletrônicos) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2015. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3142/tde-20062016-105809/>. Acesso em: 3 maio 2017.

UBIDOTS, C. **Ubidots about**. 2017. Disponível em: <https://ubidots.com/about>. Accesso em: 8 abr. 2017.

WEISER, M. The computer for the 21st century, **SIGMOBILE Mob. Comput. Commun Rev.**, New York, v. 3, n. 3, p. 3-11, jul. 1999.

WHITE, T. **Hadoop:** The Definitive Guide. 4.ed. [S.l.]: O'Reilly Media, Inc., 2015.


Cruzeiro do Sul
Educacional