

Monografia

Pós-Graduação

Bloco B

Título: O uso de Pig Latin no preparo de dados para um modelo de regressão.

Componentes: Carlos Gomes de Oliveira, Matheus Luiz Costa Freitas, Carlos Eduardo Toledo de Miranda

Orientador: Eduardo Morelli

Rio de Janeiro, 14 de novembro de 2017.

**Sumário**

**1** **Detalhamento teórico** 3

1.1 Introdução 3

1.2 Aplicações 4

1.3 Arquitetura 5

1.4 Instalação 9

1.5 Interação 10

1.6 Integração com soluções programadas em Python 12

**2** **Evidencias** 16

**3** **Conclusões** 20

**4** **Referências Bibliográficas** 21

# **Detalhamento teórico**

## Introdução

O que é Pig.

Antes de falar sobre Pig, precisamos falar sobre Hadoop, e antes deste, sobre dados.

Dados são o novo commodity do século 21, pois com dados e algoritmos é possível prever o comportamento humano, eventos econômicos, condições climáticas, projetar melhores produtos e serviços, e a lista cresce cada vez mais.

Quanto mais dados são adicionados, com ajustes nos algoritmos os resultados se tornam melhores.

Um servidor comum possui limitações no volume de dados que pode armazenar e processar simultaneamente. Uma solução é fracionar os dados e o processamento entre vários servidores.

Hadoop é um exemplo de infraestrutura para Big Data que permite gerenciar a escala de um cluster de computadores construídos com hardware de baixo custo, permitindo o crescimento do cluster caso o volume de dados armazenado ou a necessidade de processamento venha a crescer.

Esta infraestrutura permite manter o tempo de processamento constante mesmo com o aumento do volume de dados ou de processamento adicional, e inclui redundância que reduz o risco de perda de dados ou de parada do sistema em caso de falha de algum dos componentes do cluster.

Hadoop endereça o problema do tamanho com HDFS e MapReduce, sendo HDFS um sistema que armazena os dados em pequenos fragmentos distribuídos entre os discos dos servidores (nós) do cluster e MapReduce o algoritmo que realiza a leitura/atualização e processamento desses dados.

MapReduce fraciona os dados em pequenos blocos e os distribui pelos componentes de hardware chamados nós. Os dados são processados simultaneamente, e apenas serializa o processamento quando alguma agregação deve ser aplicada, como uma contagem, soma ou agrupamento, e esta é a fase Reduce.

Todos os dados armazenados possuem cópias em outros nós, assim, se um nó falha, outros nós assumem suprindo cópias dos dados que foram perdidos.

Para construir programas usando esta infraestrutura é necessário codifica em JAVA, e possuir um conhecimento muito bom do funcionamento das fases de Map e Reduce, para aproveitar os benefícios necessários da distribuição de dados.

Adicionalmente, na maior parte nós precisamos executar queries ad-hoc nos dados, e para construir programas complexos que executarão apenas uma vez é um desperdício de recursos de programação.

Pig foi concebido para endereçar estas necessidades.

Essa linguagem provê um nível de abstração acima da camada JAVA e fornece acesso a todo o poder e flexibilidade do Hadoop, sem requerer a programação de aplicações de processamento extensivo de dados em linguagem de baixo nível em JAVA.

Pig também permite a execução paralelizada de fluxos e dados no Hadoop e pode ler dados de diversas fontes, processá-las, agrega-las e armazenar os resultados num novo arquivo. Tudo em 4 a 5 linhas de código.

O nome da linguagem é Pig Latin.

## Aplicações

O caso de negócio básico do Pig é o ETL (EXTRACT, TRANSFORM, LOAD). Pois os comandos básicos realizam o processamento de terabytes de dados com poucas linhas de código Pig Latin digitadas a partir do console. O objetivo de sua criação no Yahoo! teve por objetivo permitir que pesquisadores e engenheiros de software pudessem minerar arquivos gigantescos.

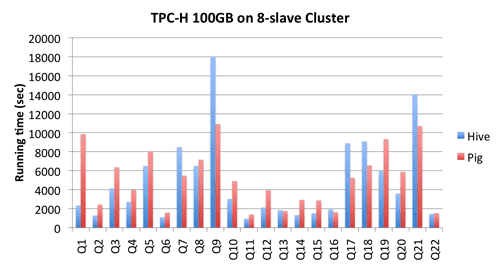
Pig habilita o programador a desenvolver como se estivesse escrevendo queries, já que provê vários comandos que permitem inspecionar as estruturas de dados à medida que vai escrevendo o código. Adicionalmente, pode processar uma amostra representativa dos seus dados para verificar se existem erros em sua execução antes de utiliza-lo no arquivo inteiro.

O processamento de dados no Pig ocorre em três fases, na primeira, um ou mais arquivos são carregados, na segunda, diversos comandos efetuam as transformações necessárias, e na terceira, um ou mais resultados de fases anteriores são armazenados novamente no sistema de arquivos Hadoop, garantindo a sua persistência.

Um ponto importante ao desenvolver aplicações em Pig é o desempenho de comandos e scripts, no website do projeto Pig no tópico de desempenho estão listadas as melhores práticas para melhorar o desempenho. No website da Horton cito um trabalho que demonstra a diferença que tais práticas fazem num comparativo de Pig com Hive. Um exemplo é a conversão de tipos, pois Pig permite avaliação preguiçosa, isto é, podemos carregar os dados e ele avalia o tipo de dados de cada coluna em tempo de carga. O “Benchmark” publicado na Horton indica que a declaração explícita em Pig, em oposição ao uso de tipos já declarados no esquema ou avaliação preguiçosa, apresenta o melhor desempenho.

Neste artigo, **Pig Performance And Optimization Analysis**, o autor compara Pig com Hive, em 22 queries similares.

O desempenho de ambas as ferramentas é misto, mas Pig pode ser otimizado em tres áreas: O uso de Map Aggregation em vez do Combiner; conversão de tipos explícita em vez de implícita; e ajustar o plano de execução para evitar a execução de jobs adicionais em Pig nas operações de Order By, em comparação com Hive.



## Arquitetura

Os scripts escritos em Pig Latin são convertidos internamente numa série de processos de MapReduce, e sua transformação ocorre de acordo com a figura abaixo.

**Os componentes do Pig são:**

Parser – por onde os scripts em Pig Latin inicialmente passam, que verifica a sintaxe do script, realiza a verificação de tipos de dados entre outras validações. O resultado do parser é um DAG (directed acyclic graph) ou grafo acíclico dirigido, estrutura comumente utilizada em compiladores para produzir um mapa de comandos, sua dependência e sequencia de execução. Esse mapa contém os comandos de Pig Latin e os operadores lógicos.

No DAG, os nós representam os operadores lógicos e os vértices representam os fluxos de dados do script.

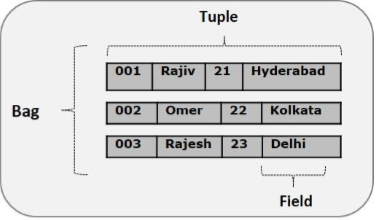


Optimizer (Otimizador) – o plano (DAG) é passado para o otimizador lógico, que realiza melhorias de desempenho lógicas como projeções e otimização pushdown, que movimenta operações ao longo do grafo para o ponto onde representem o menor custo, por exemplo, movimentar uma transformação intermediária para a fase final de carga.

Compiler (Compilador) – o plano otimizado é compilado numa série de processos (jobs) de MapReduce.

Execution engine (Máquina de execução) – finalmente os jobs são executados no Hadoop a partir de uma lista ordenada para gerar os resultados esperados. Este passo funciona como uma agendada de operações (scheduler).

**Modelo de dados:**



Qualquer valor escalar ou caractere simples em Pig Latin, seja qual for o tipo de dado associado, é conhecido como átomo (Atom). Ele é armazenado como caractere e pode ser usado como caractere (string) ou número. Os tipos atômicos de Pig são: int, long, float, double, chararray, e bytearray. Um fragmento de dado ou um valor atômico simples é chamado de campo (field).

A Tupla (Tuple) é um registro formato por um conjunto ordenado de campos, os quais podem ser de qualquer tipo. Uma tupla é similar a uma linha numa tabela de dados de um RDBMS (Sistema de Banco de Dados Relacional).

Um conjunto desordenado e tuplas é um **Bag** Representa uma coleção de tuplas (que podem ser não únicas). As tuplas dentro de um bag não possuem necessariamente a mesma ordem, tipo, ou quantidade em cada linha. Delimilata por **{ }**.

Um mapa (map ou data map) é um conjunto de pares chave-valor. A chave deve ser do tipo chararray e deve ser única. O valor pode ser de qualquer tipo. Delimitado por **[ ]**.

Uma relação (relation) é um bag de tuplas e podem ser desordenadas, pois não há garantia de que as tuplas sejam processadas ou recuperadas em alguma ordem específica.

Tez – termo hindu para velocidade que nomeia um componente do ecossistema Hadoop que prove um framework de uso geral altamente flexível de uso geral, que cria e simplifica tarefas de carga em Hadoop tanto em pequena escala (baixa latência) como em alta escala (elevado volume). O Tez generaliza o paradigma de MapReduce para um framework mais poderoso ao prover o Hadoop com a habilidade de otimizar e executar um complexo DAG de tarefas para um único job, projetos do ecossistema Apache Hadoop tal como Hive, Cascading, e no nosso caso o Pig podem se beneficiar ao operar com o Tez.

Um modelo básico de processamento em Pig tem 3 fases.

* + 1. Os dados são carregados de uma ou mais fontes utilizando o comando LOAD
    2. Várias operações de transformação podem ser realizadas com comandos de:
       1. Projeção de colunas e tipos de dados com loops FOREACH
       2. Seleção de dados relevantes com o FILTER
       3. Merge de vários arquivos carregados com JOIN
       4. Agregação com os comandos GROUP/ORDER BY e funções associadas
    3. Finalmente os arquivos processados podem ser persistidos novamente no Hadoop com o comando STORE.

Ao contrário do SQL, o Pig pode:

* Utilizar avaliação de tipos preguiçosa ( durante e após a carga)
* Realizar ETL (extract, transform and load)
* Pode armazenar dados em qualquer ponto do script
* Declara os planos de execução
* Suporta o Split de fluxos de dados, permitindo que o processamento ocorra em paralelo no DAG, em vez de seguir um fluxo sequencial
* Na fase de load podem ser carregados dados estruturados (tipados) ou não estruturados. Dados não estruturados podem ser estruturados implicitamente ou explicitamente.

Por padrão o Pig executa scrips em modo Multiquery.  No modo Multiquery o script todo é primeiro analisado em suas partes para ver se as tarefas podem ser combinadas para reduzir a quantidade total de trabalho a ser executado. A execução só inicia após toda a análise (parsing) estar completa.

Para desligar o modo multiquery, executar o script na ordem das tarefas escritas, e não usar otimizador, deve-se passar o parâmetro -no\_multiquery ou -M. O uso de DUMP também desliga o otimizador.

pig -no\_multiquery foo.pig

## Instalação

Antes de instalar Pig, para usufruir dos benefícios da arquitetura de dados distribuídos, é necessário instalar o Hadoop.

Após ter instalado o Hadoop com sucesso, deve-se baixar a versão mais adequada, geralmente a última versão estável, que pode ser obtida no website do projeto Apache Pig (<http://pig.apache.org/releases.html>).

Pig não precisa ser instalado diretamente na mesma máquina em que se instalou o Hadoop, podendo ser instalado em máquinas que apenas executam os jobs de MapReduce, apesar de poder ser instalado num desktop ou laptop como uma aplicação cliente, os administradores do ambiente Hadoop geralmente o instalam em máquinas que não são nós de dados ou tarefas, e funcionam com um portal para acesso ao ambiente.

Essa estratégia facilita a administração do ambiente, e permite um controle mais rigoroso no acesso aos dados, aumentando a segurança e a estabilidade do ambiente, facilitando também a sua manutenção.

As máquinas onde o Pig é instalado são chamadas de Gateway Machines, ou máquinas de portal. Por ser escrito em JAVA ele é portável para vários sistemas operacionais, mas Hadoop precisa de Linux para o seu sistema de arquivos distribuído. Assim os desenvolvedores geralmente utilizam algum sabor de Linux ou Mac OS para o desenvolvimento.

Os passos para a correta instalação do Pig são:

1 - Ter em mente que Pig é uma aplicação que opera em modo cliente.

2 – Verificar se JAVA está instalado e na versão correta.

2 - Realizar o download de uma versão estável do site do projeto apache.

<http://pig.apache.org/releases.html>

3 - Realizar o unzip e untar em local adequado, que pode ser sua estação de trabalho ou uma maquina de portal.

$ gunzip pig-0.17.0.tar.gz

$ tar xzf pig-0.17.0.tar.gz tar

4 – Adicionar o caminho dos binários do Pig (/bin) à variável de path.

5 – Exportar as variáveis de ambiente

$ export PIG\_HOME=XXX (caminho para o home do Pig)

$ export PATH (caminho para os binários do Pig)

$ JAVA\_HOME=XXX (caminho do home do JAVA)

Outra forma de utilizar Pig é através de pacotes da Horton ou da Cloudera, que possuem instalações completas de Hadoop, Pig e muitos outros componentes do ecossistema Hadoop. Esse método tem a facilidade de uso imediato tanto para teste, como para produção, sem o envolvimento com a instalação de cada componente individual, por outro lado, o suporte e as atualizações devem ser realizados por meio dos fornecedores, e o suporte ao pacote possui um custo embutido.

Pig também pode ser instalado sem outros componentes do Hadoop, a única dependência é ter o Java instalado.

Isto permite executar scripts em Pig Latin ou executar seus comandos no grunt em modo local.

Esta é uma forma de se debugar e testar scripts Pig sem estar conectado a um cluster Hadoop.

Instalação (para a versão 0.17, atualizar o atalho para novas versões):

**$ wget http://www-us.apache.org/dist/pig/pig-0.17.0/pig-0.17.0.tar.gz**

**$ -rw-rw-r-- 1 carlos carlos 230606579 Jun 20 06:31 pig-0.17.0.tar.gz**

**$ tar -xzf pig-0.17.0.tar.gz**

**$ drwxr-xr-x 16 carlos carlos 4096 Jun 2 2017 pig-0.17.0**

Localizar o home do Java, neste caso já estou com o Java 9

**$ whereis java**

**$ java: /usr/bin/java /usr/share/java /usr/share/man/man1/java.1.gz**

**$ ls -alrt /usr/bin/java**

**$ lrwxrwxrwx 1 root root 22 Nov 21 2016 /usr/bin/java -> /etc/alternatives/java**

**$ ls -lart /etc/alternatives/java**

**$ lrwxrwxrwx 1 root root 42 Mar 29 2017 /etc/alternatives/java -> /usr/lib/jvm/java-9-openjdk-amd64/bin/java**

**$ ls -lart /usr/lib/jvm/java-9-openjdk-amd64/bin/java**

**$ -rwxr-xr-x 1 root root 6480 Abr 14 2016 /usr/lib/jvm/java-9-openjdk-amd64/bin/java**

Adicionar ao final de .bashrc

**# Set PIG\_HOME**

**export PIG\_HOME=/home/carlos/pig-0.17.0**

**export PATH=$PATH:/home/carlos/pig-0.17.0/bin**

**export PIG\_CLASSPATH=$HADOOP\_CONF\_DIR**

**export JAVA\_HOME=/usr/lib/jvm/java-9-openjdk-amd64**

## Interação

Podem-se executar comandos de Pig Latin de duas formas, interativa ou em lote (script).

A forma de execução pode ser tanto local, acessando apenas os arquivos presentes na própria máquina que invocou pig, quanto em modo MapReduce, que é o default.

Ao invocar pig de forma interativa aparece o prompt de comando chamdo Grunt. Neste prompt pode-se escrever, corrigir, e executar comandos Pig, linha a linha.

Normalmente em instalações de teste executa-se o pig em modo local e interativo, utilizando a linha de comando:

$ pig -x local

Em seguida surge o prompt de comando:

$ grunt>

Para sair digita-se quit ou Crtl+D.

A outra forma de invocar o prompt do Grunt é eliminar o parâmetro –x local, que invoca o Grunt em modo MapReduce.

$ pig

$ grunt>

Pode-se também executar comandos shell de Hadoop diretamente do Grunt, bastando-se prefixar cada comando com a palavra chave fs.

$ grunt> fs –ls

Após testar os comandos em modo interativo, os mesmos podem ser armazenados em arquivo texto e executados em lote com o comando abaixo, que por exemplo executa um script chamado teste01.pig em modo local.

$ pig -x local -script teste01.pig

Programas Pig também podem ser executados de forma embutida, a partir do JAVA usando a classe PigServer, é similar a execução de comandos SQL utilizando JDBC.

Não importa como os programas Pig são executados, seja utilizando scripts, grunt ou embutidos, também é possível especificar o modo de execução.

Existem 3 modos, Local, MapReduce e Tez:

Em modo local Pig executa uma única JVM e apenas acessa o sistema de arquivos local.

Em modo MapReduce, Pig traduz os comandos em jobs de MapReduce e os executa num cluster Hadoop.

Em modo Tez, Pig funciona em modo MapReduce, mas utiliza o framework Tez para otimizar o processamento e acelerar as execuções.

Finalmente, outra possiblidade de se escalar o uso da linguagem Pig Latin é com o uso de UDF, que são funções definidas pelo usuário. UDFs serão exploradas em mais detalhe no próximo tópico.

É possível realizar a passagem de parâmetros a partir da linha de comando, deste modo scripts em Pig podem executar processos de MapReduce flexíveis pois o mesmo código passa a atender diversas situações definidas pelos parâmetros passados. Por exemplo, pode-se usar parâmetro para definir o valor de um filtro, assim como foi realizado neste trabalho, para que o mesmo script pudesse gerar uma saída de processamento para cada estação meteorológica da França.

No exemplo abaixo vamos filtrar os arquivo de estação para os registros com stid (identificador de estação) com valor ‘33’. Na linha de comando a passagem de parâmetro usa um alias (id\_estacao) associado a um valor (33), usando a seguinte sintaxe **-param <alias>=<valor>**.

**pig -f temperatura\_estacao.pig -param id\_estacao=33**

Para usar o valor do parâmetro no código basta usar a sintaxe $<alias do parâmetro>, como no destaque do exemplo abaixo.

**t1 = LOAD '/user/horton/infnet/temperatura/stations.txt'**

**USING PigStorage(',')**

**AS(stid:int, estacao:chararray);**

**t2 = FILTER t1 BY stid == $id\_estacao;**

## Integração com soluções programadas em Python

Funções definidas pelo usuário ( UDF ) são usadas para estender e customizar as funcionalidades das funções pré-definidas (built in functions) no PIG. UDFs podem estender as funcionalidades de Eval, Load, Store, Math, String, Bag e Tuples Functions do PIG e podem ser codificadas em Java, Python e Javascript.

Neste trabalho foi necessário usar esta extensão de funcionalidade para gerar a partir de uma data, formada por ano, mês e dia, o dia da semana. Desta forma poderia ser percebida a assinatura da variação de consumo de energia pelo dia da semana e esta informação poderia ser incorporada nas análises para determinação da localização do domicilio. Usando esta UDF foi possível perceber um aumento do consumo na quarta-feira, que tornava este dia singular em relação aos demais dias úteis da semana. Ligamos inicialmente esta observação ao domicílio estar localizado na França, uma vez que neste país as escolas funcionam em horário integral com exceção da quarta-feira quando funcionam apenas em meio período, ou seja, na quarta-feira com as crianças com idade escolar passam mais horas em casa e aumentam o consumo de energia. Posteriormente esta suposição confirmou-se através da análise dos metadados do dataset.

Como havia a necessidade da criação de uma UDF aproveitou se para criar esta função em Python para atingir um dos objetivos deste trabalho, usar uma ferramenta do ecossistema do Hadoop integrando esta ferramenta com Python. Por mais simples que esta função em Python seja, ela demonstra esta integração entre PIG e Python.

Em comparação com as UDFs desenvolvidas em Java, as UDFs desenvolvidas em Python e Javascript, expões apenas a interface de métodos ou funções básicas. As interfaces Algebraic e Accumulator não estão implementadas em Python ou Javascript, deste modo funções UDF nestas linguagens estão restritas a transformações simples de valores e não podem ser usadas em funções de load, restores, e intrínsecas ao MapReduce. Em tempo de execução o PIG automaticamente detectará o uso de funções UDF e carregará para o backend o correspondente Jython (implementação de Python para a plataforma Java ) ou Rhino (Javascript implementado em Java para uso “embebded” em Java) jar que encapsula o script em Python ou Javascript.

Neste caso, de transformação de data para dia da semana, UDFs desenvolvidas com a interface básica, que expõem funções Eval são suficientes para o requisito de transformação de atributo em GENERATE de FOREACH.

A função em Python usa a sintaxe do Python com o auxílio de um decorator para indicar ao PIG o tipo do atributo de retorno da função. Segue a função desenvolvida em Python para a transformação de ano, mês e dia em dia da semana.

O decorator (contêiner ou wrapper) @outputschema é usado para decorar, estender comportamento, à função UDF atribuindo um schema no retorno da função. O decorator é uma função que recebe outra como parâmetro e estende o comportamento inicial.

**import datetime**

**@outputSchema('diadasemana:int')**

**def data2diadasemana(ano, mes, dia):**

**return (datetime.datetime(ano, mes, dia)).weekday()**

A função recebe três parâmetros (ano, mês e dia) e retorna o dia da semana como um inteiro, sendo segunda-feira definida por 0 e domingo por 6.  O decorator @outputschema, informa o tipo do retorno da função ao PIG.  Este tipo fará parte dos schemas do PIG.

Para usar a função no script PIG é necessário o registro da função, para isto foi adicionado ao script PIG a seguinte linha:

**REGISTER 'diadasemana.py' using jython AS udfpy;**

Após o registro da função basta chamar o arquivo python registrado por seu alias (udfpy, neste caso). Função definida do arquivo python. Como no trecho de código abaixo.

hpc3 = FOREACH hpc2 GENERATE

      ((int)$2, (int)$1, (int)$0, (int)$3, (int)$4,

**(int)udfpy.data2diadasemana((int)$2, (int)$1, (int)$0)**,

      (float)$6, (float)$7, (float)$8,

      (float)$9, (float)$10, (float)$11, (float)$12,

      (float)$6\*1000/60 - (float)$10 - (float)$11 - (float)$12,

      (float)$6\*1000/60 )

      AS ((ano:int , mes:int, dia:int, hora:int,  minuto:int, diasemana:int,

      activpwr:float, reactpwr:float, voltagem:float, intensidade:float,

      cozinha:float, lavanderia:float, aquecimento:float, outros:float, total:float));

Neste exemplo hpc3 conterá em seu schema com um atributo no formato diadasemana:int.

**Comentários**

As transformações de projeção de colunas e tipos de dados com loops FOREACH e GENERATE podem ser de complexas na escrita caso tenham muitas transformações e aplicação de funções. Não é apenas o caso de legibilidade (facilidade de leitura) e redigibilidade (facilidade de escrita), nestes casos o que foi observado é que o resultado de um FOREACH muito complexo com muitos atributos com alta redigibilidade (escrito em uma linha apenas de FOREACH) pode não ser o esperado e encontrar o erro neste cenário de baixa legibilidade é muito difícil. Considerando o uso do otimizador do PIG é mais adequado a escrita de vários FOREACH em sequência cada um tratando de um grupo determinado de transformações, com perda da redigibilidade, no entanto com mais clareza de legibilidade. Deste nodo encontrar problemas na geração do resultado esperado é mais fácil, inclusive pela possibilidade de poder inspecionar o resultado a cada passo, ao mesmo tempo que o otimizador garantirá uma performance adequada.

Observe no exemplo abaixo a sequência de FOREACH em m5, m6 e m7. Inicialmente a ideia seria programar a transformação de projeção em uma única linha, no enteando, somente ao separar as linhas em distintos FOREACH foi possível chegar ao resultado desejado sem erros.

m1 = LOAD '/user/horton/infnet/temperatura/TG\_STAID\*.txt'

USING PigStorage(',')

AS(scid:int, data:chararray, temperatura:int, qualidade:int);

m2 = FILTER m1 BY scid is not null and qualidade == 0 and data > '20061200' and data < '20101200';

m3 = JOIN est BY scid, m2 BY scid;

m4 = GROUP m3 BY data;

**m5 = FOREACH m4 GENERATE FLATTEN(group),AVG(m3.temperatura);**

**m6 = FOREACH m5 GENERATE SUBSTRING($0,0,4), SUBSTRING($0,4,6), SUBSTRING($0,6,8), $1;**

**m7 = FOREACH m6 GENERATE $0,$1,$2, udfpy.data2diadasemana((int)$0, (int)$1, (int)$2), $3;**

medicoes\_ordenadas = ORDER m7 BY $0,$1,$2;.

# **Evidencias**

**Scripts**

Para a execução dos scripts basta chamar o pig e o script que deseja executar:

pig consumo\_total\_mediadiaria.pig

pig temperatura\_mediadiaria.pig

Para a execução do script por estação deve-se passar um parâmetro do id da estação. O exemplo a seguir executa o script para a estação de BOURGES, de id 33.

pig -f temperatura\_estacao.pig -param id\_estacao=33

Sem passar o -f o script não considera o parâmetro.

**Resultados**

O diretório de resultados contém os arquivos gerados pelos processos de map reduce executados pelo Pig obtidos para o diretório local pelo comando "hadoop fs -get <nome do diretório>"

Para abrir os arquivos de resultados vá para a pasta do resultado desejado e abra o arquivo part-r-00000, exemplo:

resultados/<nome do resultado>/part-r-00000

O arquivo é texto com registros separados por ;.

**Consumo médio diário**

Registros ordenados, separados por **;** contendo:

ano,

mês,

dia,

dia da semana (0-6, sendo 0 segunda-feira),

média diária do consumo total por minuto,

soma do consumo diário por minuto

**Temperatura média do dia**

Não conseguimos gerar uma saída com todas as estações, deste modo geramos uma saída por estação. Acreditamos que comparando o consumo diário com as curvas de temperatura de cada uma das estações poderemos inferir de quais ou quais estações o dado do consumo tem comportamento mais adequado. Descobrindo assim de qual região da França são os dados de consumo.

Registros ordenados, separados por **;** contendo:

ano,

mês,

dia,

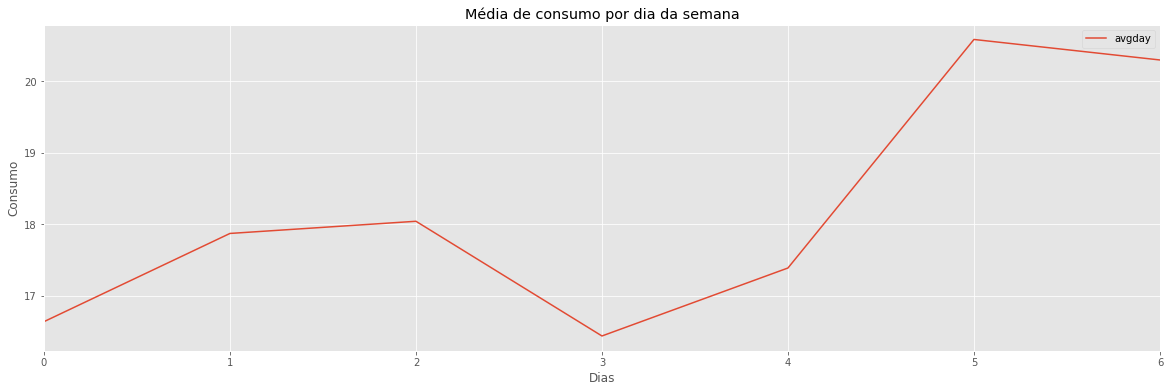
dia da semana (0-6, sendo 0 segunda-feira),

temperatura média diária (x10), mantive o dados como no original que considera a temperatura x10 para trabalhar com inteiros.

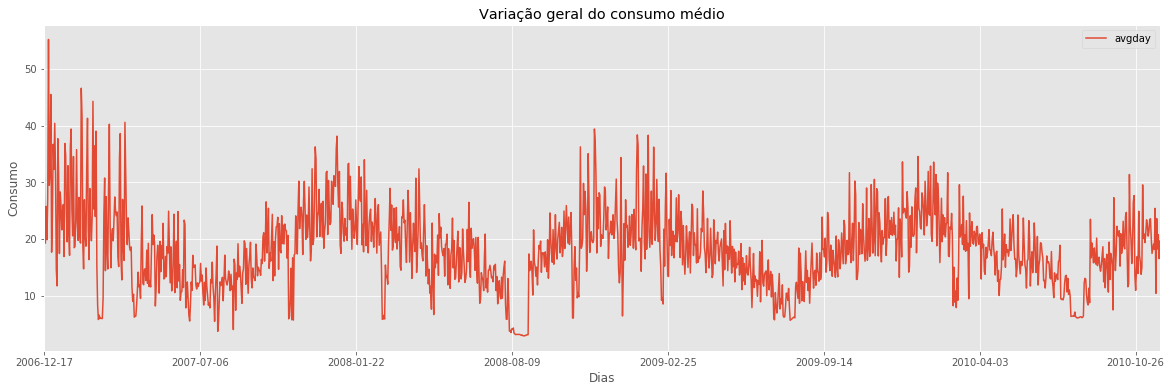
O arquivo em **temperatura\_mediadiaria** contém a temperatura média do dia considerando todas as estações, os arquivos **temperatura\_estacao\_<Id Estacao>** contém a temperatura média do dia considerando apenas a estação referida pelo identificador.

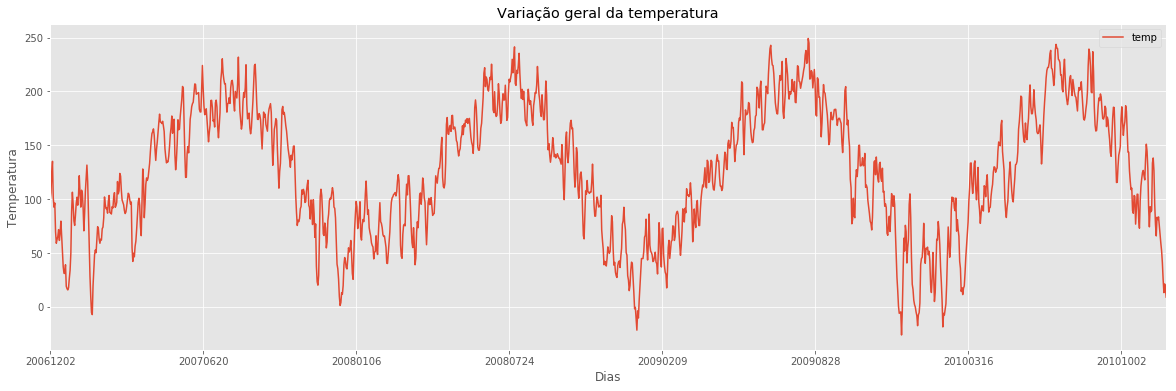
Um aspecto interessante a ser destacado são as dificuldades e limitações na elaboração de cada um dos 4 entregáveis.

No gráfico abaixo obtido sobre um dos arquivos gerados, podemos observar que um dos dias da semana (quarta-feira), possui o maior consumo médio, seguido de sábado e domingo.

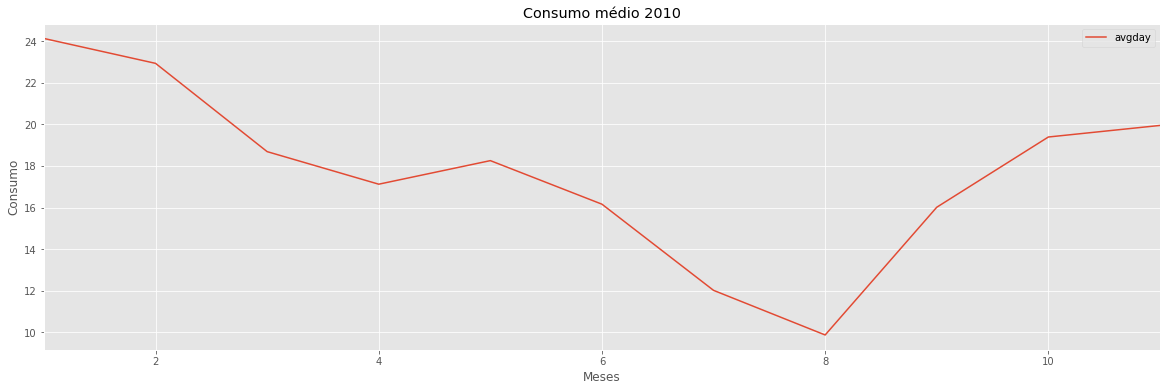


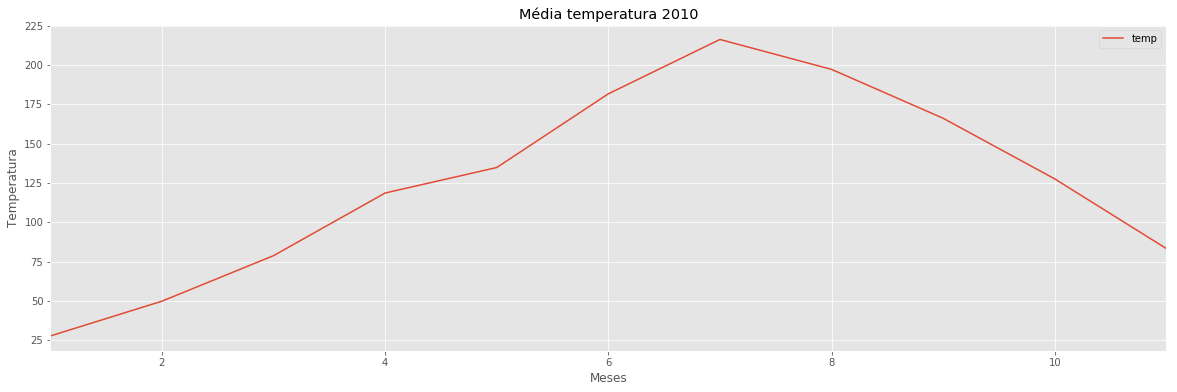
Aqui podemos comparar a variação do consumo e da temperatura em 2010, em valores reais, ponto a ponto, podemos observar que há uma clara tendência geral de crescimento e queda, apesar de fortes variações de prazo mais curto.





Ao utilizar valores médios, a correlação fica bem mais clara entre variação de temperatura e variação de consumo, com isso, os dados médios podem ser utilizados para treinar um modelo de regressão para projetar o consumo médio com base na previsão futura de temperatura.





# **Conclusões**

Eu percebo uma maior flexibilidade no Pig em comparação com o Hive, é quase como se estivesse programando diretamente no plano de acesso da query, adicionalmente, em Pig, podemos executar fluxos de dados que seriam impossíveis em SQL, ou pelo menos num único SQL, com a vantagem de ter os mesmos fluxos otimizados para execução paralela.

A possibilidade de executar o Pig localmente no Ubuntu é um grande acelerador, também é possível instalar no Windows, testamos várias sugestões, mas todas falharam. Somente tive sucesso em Linux Ubuntu.

Jython ainda não foi atualizado para o Java 9, se for necessário, deve-se alterar o path do JAVA\_HOME apontando-o para o Java 8 e incluí-lo no início do PATH em seguida.

# **Referências Bibliográficas**

GATES, Alan **Programming Pig – Dataflow Scripting with Hadoop.** O’Reilly Media Inc., 2011.

White, Tom **Hadoop: The Definitive Guide.** O’Reilly Media Inc., 2017.

UCI Machine Learning Repository **Energy Dataset** Disponível em: https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption# Acesso em: 05/12/2017

European Climate Assessment & Dataset Project **Weather DS** Disponível em: http://www.ecad.eu/dailydata/index.php Acesso em: 05/12/2017

Hortonworks Tutorial **What is Tez?** Disponível em: https://br.hortonworks.com/tutorial/how-to-process-data-with-apache-pig/#what-is-tez Acesso em: 05/12/2017

Hortonworks Blog **Pig Performance And Optimization Analysis** Disponível em: <https://br.hortonworks.com/blog/pig-performance-and-optimization-analysis/> Acesso em: 05/12/2017

Apache Hadoop Project **Apache Pig Releases** Disponível em: <http://pig.apache.org/releases.html> Acesso em: 05/12/2017

Wikipedia **Pig (programming tool)** Disponível em: <https://en.wikipedia.org/wiki/Pig_(programming_tool)> Acesso em: 05/12/2017

Apache Hadoop Project **Pig Documentation: Performance and Efficiency** Disponível em: <https://pig.apache.org/docs/r0.9.1/perf.html> Acesso em: 05/12/2017

Tutorials Point **Apache Pig - Architecture** Disponível em: <https://www.tutorialspoint.com/apache_pig/apache_pig_architecture.htm> Acesso em: 05/12/2017

Edureka! **Instalação de Apache Pig em modo local no Linux**. Disponível em: <https://www.edureka.co/blog/apache-pig-installation> Acesso em: 05/12/2017

Wikipedia **DAG (Directed Acyclic Graph)** Disponível em: <https://en.wikipedia.org/wiki/Directed_acyclic_graph> Acesso em: 14/12/2017

Stackoverflow **Difference between df-where and df-df in pandas python.** Disponível em: https://stackoverflow.com/questions/40332974/difference-between-df-where-and-df-df-in-pandas-python Acesso em: 14/12/2017

Blog **Como criar seu primeiro projeto de data science.** Disponível em: https://paulovasconcellos.com.br/como-criar-seu-primeiro-projeto-de-data-science-parte-2-de-2-cb9a2fe05eff Acesso em: 14/12/2017

Stackoverflow **Pandas writing dataframe to csv file.** Disponível em: https://stackoverflow.com/questions/16923281/pandas-writing-dataframe-to-csv-file Acesso em: 14/12/2017