

Monografia

Pós-Graduação

HAWQ

Título: Estudo de caso – Banco NoSQL HAWQ

Autor: Daniel Coelho / Fernando Peres / Marcelo Fontenele / Rosane Ricciardi

Orientador: Eduardo Morelli

Rio de Janeiro, Junho de 2016

**Sumário**

**1** **Introdução** 4

**2** **Aplicações** 6

2.1 Alto desempenho 6

2.2 100% compatível com SQL *ANSI* 7

2.3 Framework de aprendizado de máquina integrado (MADLib) 7

2.4 *Deployment* é flexível 7

**3** **Arquitetura** 9

3.1 Serviço de catálogo (*Catalog service*) 10

3.2 Servidor mestre (HAWQ *Master*) 10

3.3 Servidor em modo de espera (*Standby Master*) 11

3.4 Segmentos (*Segments*) 11

3.5 HAWQ *Network Interconnect* (Interconexão de rede) 12

3.6 Acesso a sistemas externos / *Pivotal eXtension Framework* (PXF) 12

**4** **Instalação** 14

4.1 Iniciando o HAWQ 15

**5** **Interação** 17

5.1 SELECT 17

5.2 INSERT 17

5.3 UPDATE 18

5.4 DELETE 18

5.5 CREATE TABLE 18

5.6 DROP TABLE 18

**6** **Tarefas Administrativas** 19

6.1 Diferenças entre gpfdist e PXF 19

6.2 Uso do pg\_dump e pg\_restore 20

*6.3* Estratégias de *back up* 21

6.4 Estimava de espaço requerido 21

6.5 Utilitário gpfdist 22

Figura 6.5.1: ambiente com múltiplas instâncias de gpfdist. 23

Figura 6.5.2: ambiente com um instância gpfdist em cada *segment host*. 23

6.6 Usando gpfdist para fazer back up 24

6.7 Restaurando um back up com gpfdist 25

6.8 Usando PXF para fazer back up 26

6.9 Restaurando um back up com PXF 27

6.10 Passos básicos de back up e restauração 28

**7** **Evidências** 29

7.1 SELECT 29

7.2 INSERT 30

7.3 DELETE (TRUNCATE) 30

7.4 CREATE TABLE 30

7.5 DROP TABLE 30

**8** **Conclusões** 31

8.1 Quadro comparativo 31

8.2 Integração 32

8.3 Curva de aprendizagem 33

8.4 MADLib 33

**9** **Referência Bibliográfica** 34

# **Introdução**

Neste presente trabalho de conclusão de bloco, serão apresentadas as análises sobre o HAWQ, um banco de dados noSQL que conta com um motor de consultas SQL que faz uma interface com o Hadoop. O nome HAWQ é um acrônimo e que significa HAdoop With Query, traduzindo para português Hadoop com Query (SQL). Ele foi projetado para usar SQL como linguagem de consulta de dados. Desta forma, permite o uso de uma linguagem de consulta (SQL) amplamente conhecida no mercado e que facilita a adoção da ferramenta, pois não exige a necessidade de se aprender uma nova linguagem de consulta. Essa característica, também poderia facilitar a escolha da ferramenta frente outras opções, pois como ponto positivo reduziria a curva de aprendizagem.

O HAWQ usa o HDFS (Hadoop *Distributed File System* / Sistema de arquivos distribuídos do Hadoop) como camada de armazenamento, permitindo a execução de consultas de dados (via SQL) na escala de petabytes e em alta velocidade. Ele lê e escreve nativamente no HDFS. Ele utiliza Processamento Paralelo Massivo (MPP - *Massive Parallel Processing*), característica que herdou do Greenplum (outro banco de dados NoSQL). O MPP se caracteriza pelo gerenciamento de dados que são particionados em múltiplos servidores e cada segmento se encarrega de gerenciar uma parte distinta do conjunto total de dados.

Características do HAWQ:

* Permite instalação local ou na nuvem;
* Suporte robusto a ANSI SQL: SQL-92, SQL-99, SQL-2003, extensão OLAP;
* Alta performance. Muitas vezes mais rápido que outros *engines* SQL do Hadoop;
* Otimizador paralelo de alto desempenho;
* Suporte a transações e garantia de consistência ACID;
* Fluxo de dados dinâmico através de interconexões UDP de alta velocidade;
* Suporte a múltiplos níveis de particionamento e tabelas particionadas por intervalos e listas;
* Suporte a múltiplos métodos de compressão de dados: snappy, gzip, quicklz, RLE;
* Suporte a diferentes linguagens de programação para definição de funções: python, perl, java, c/c++, R;
* Funcionalidades de data mining e *machine learning* usando **MADLib**.
* Expansão dinâmica dos nós em segundos;
* O mais avançado gerenciador de recursos em níveis: Integração com YARN e hierarquia de filas de recursos;
* Fácil acesso a todos os tipos de dados do HDFS e dados externos ao sistema como **HBase**, etc.
* Nativo do **Hadoop**: do armazenamento (HDFS), gerenciamento de recursos (YARN) até o monitoramento (Ambari);
* Autenticação e autorização: Kerberos, SSL e acesso baseado em papéis;
* Biblioteca avançada em C/C++ para acesso ao HDFS e YARN: libhdfs3 & YARN;
* Suporte a maioria das ferramentas de terceiros: Tableau, SAS, etc.
* Conectividade padrão: JDBC/ODBC.

A seguir serão apresentados os resultados deste estudo, ressaltando as características e uso possíveis do HAWQ.

# **Aplicações**

O HAWQ permite diversos tipos de aplicações, principalmente as que requerem alto desempenho nas consultas e que necessite processamento paralelo massivo de dados. Dentre os seus principais motivos que encorajam a sua utilização, caberia salientar as seguintes:

* Alto desempenho em análises avançadas
* 100% compatível com o padrão SQL ANSI
* Framework de aprendizado de máquina integrado (MADLib)
* Desenvolvimento flexível
* Opções de otimização de consultas SQL query
* Possui extensões (PXF) que permitem interações com dados armazenados em outros serviços como HBase, Hive, Pig, etc que também residem no HDFS
* Conectividade ODBC e JDBC

## Alto desempenho

O HAWQ foi projetado para permitir processamento massivo de dados e promover um alto desempenho na velocidade de resposta de consulta. Para viabilizar o alto desempenho ele foi desenvolvido em C e C++ em Linux, permitindo que as instruções estivessem mais próximas do sistema operacional, sem a necessidade de interpretadores que consomem tempo. Desta maneira, o HAWQ apresenta um altíssimo desempenho quando comparados com as outras ferramentas, sobretudo quando se compara as demais ferramentas de consulta de dados no Hadoop. O HAWQ possibilitou a melhoria de mais de 30x na performance de SQL on Hadoop e menor tempo para *insight.* Inclusive, de maneira massiva e escalável a petabytes.

Nesse sentido, os principais benefícios obtidos pelo HAWQ são os seguintes:

* Latência próxima a tempo-real;
* Permite consultas complexas;
* Habilita analytics avançada em escala.

## 100% compatível com SQL *ANSI*

O HAWQ é baseado no Postgres e desta forma é 100% compatível com SQL ANSI (*American National Standards Institute*), suportando SQL '92, '99, 2003 OLAP no Hadoop. Ele Suporta consultas complexas com *subqueries* correlacionadas, funções de janela e diferentes agregações, *joins* e cubos. Também permite uma ampla gama de funções escalares e agregações.

Os usuários também podem se conectar a HAWQ via ODBC e JDBC. O benefício para as empresas reside na existência de um grande ecossistema de inteligência de negócios, análise de dados e ferramentas de visualização de dados que trabalham com HAWQ fora da caixa devido o seu suporte SQL totalmente compatível. Além disso, aplicativos analíticos escritos no HAWQ são facilmente portáveis para outros motores de consultas SQL e vice-versa.

## Framework de aprendizado de máquina integrado (MADLib)

O HAWQ possui um módulo integrado para realizar *analytics* avançado para BigData. A Madlib é uma biblioteca *open-source* baseada em Postgress, com diversas funcionalidade de analytics avançado e foi desenhada para interagir principalmente com Postgres, Greenplum e HAWQ. Esta poderosa biblioteca possibilita a implementação de métodos matemáticos, estatísticos e de aprendizado de máquina (*Machine Learning*) em banco de dados. Permite também alto desempenho com quantidades massivas de dados.

## *Deployment* é flexível

O HAWQ possui – tanto para o HDP e quanto para o Pivotal HD – um gerenciamento relativamente fácil via Ambari. O Apache Ambari é uma ferramenta projetada para facilitar a gestão Hadoop, torná-la mais simples através do desenvolvimento de software para o provisionamento, gerenciamento e monitoramento clusters de Apache Hadoop.

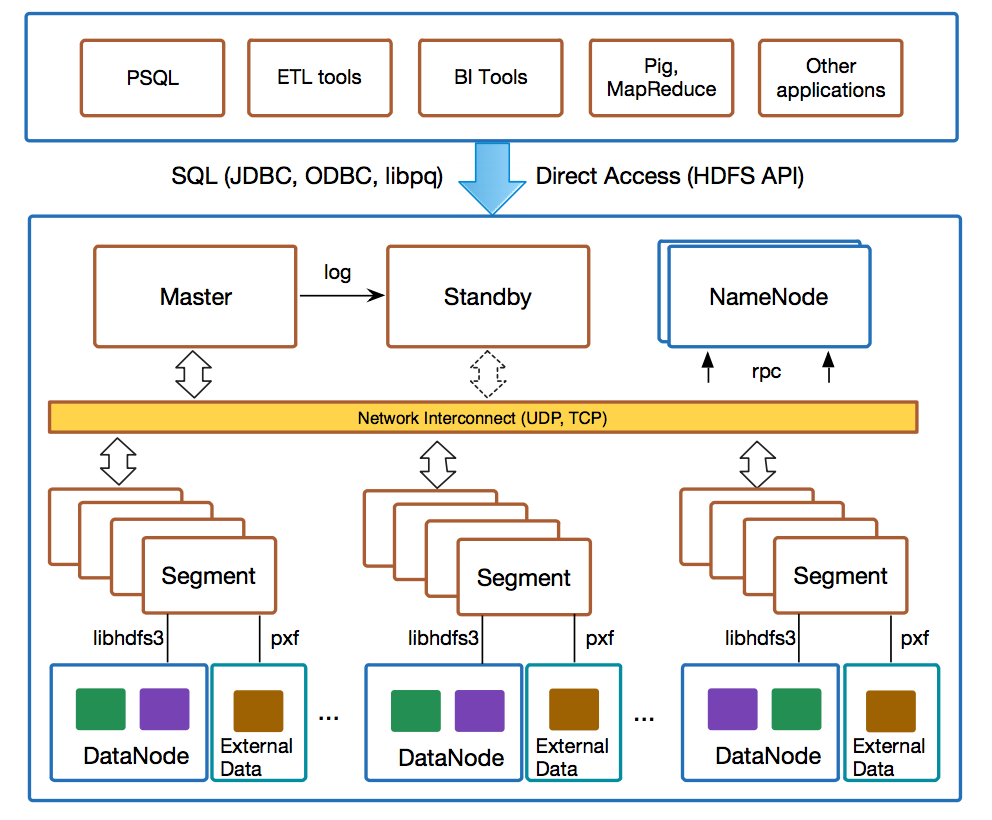
O Ambari fornece uma interface intuitiva utilização de fácil gerenciamento Hadoop web Ui. Além disso, suporta múltiplos formatos Hadoop, tais como: Parquet, Avro, JSON, ORC e outros. O HAWQ também possui conectores para disponibilizar dados HAWQ para outras ferramentas SQL, ampliando a possibilidade de ferramentas e flexibilidade de uso da ferramenta.

Nesse sentido, os principais benefícios obtidos pelo HAWQ são os seguintes:

* Flexibilidade;
* Accessibilidade;
* Portabilidade.

# **Arquitetura**

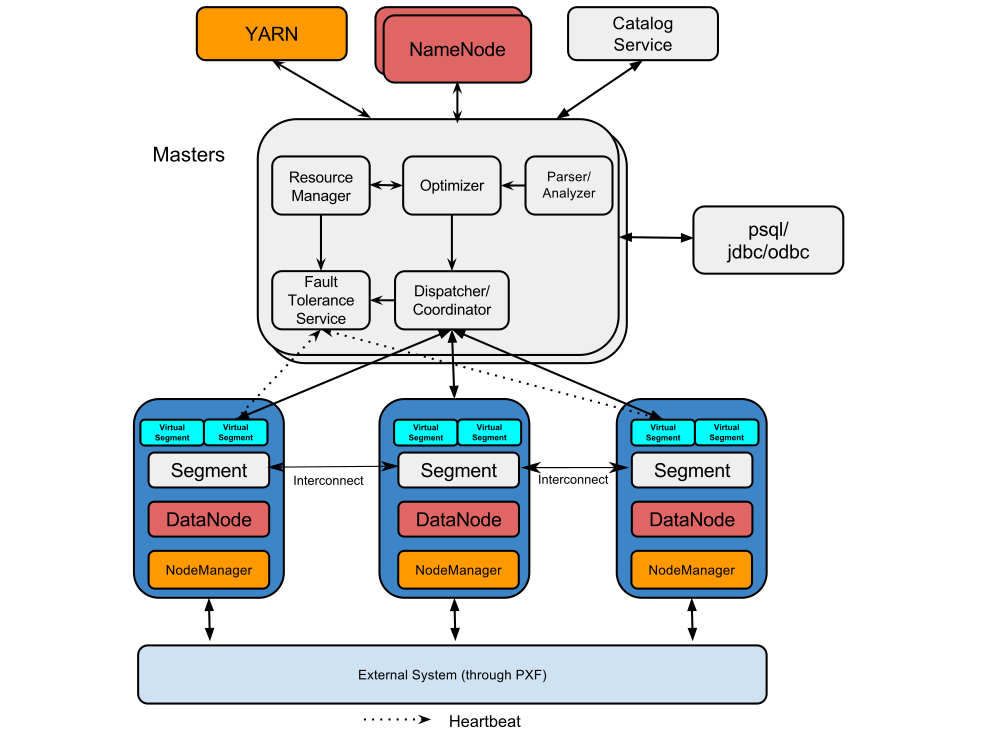
O HAWQ possui uma arquitetura em camadas e foi projetado para ser um mecanismo de processamento massivo e paralelo de dados, com SQL otimizado e de alto desempenho. A arquitetura do HAWQ foi desenhada para permitir alto desempenho. Ele utiliza uma técnica de dividir para conquistar, quebrando consultas complexas em pequenas tarefas, para distribuí-las, permitindo realizar diversas consultas nas unidades de processamento simultaneamente para a execução das consultas de dados em alta velocidade.



**Figura 3.1:** Visão 1: simplificada da Arquitetura do HAWQ

Como configuração mínima em ambiente de produção possui quatro componentes básicos:

* Servidor mestre (HAWQ *Master*)
* (*Standby Master*)
* HDFS *Name nodes*
* Segmentos (*Segments*)



**Figura 3.2:** Visão 2: Arquitetura do HAWQ

A seguir, serão descritos os principais elementos da arquitetura do HAWQ.

## Serviço de catálogo (*Catalog service*)

O serviço de catálogo do HAWQ armazena e mantem todos os metadados. Os metadados são formados por:

* Informações das tabelas – schemas, nomes, arquivos;
* Estatísticas – por exemplo, número de valores únicos, intervalo de valores, exemplos;
* Informações sobre usuários, grupos, prioridades, etc.

## Servidor mestre (HAWQ *Master*)

É uma instância modificada do banco de dados Postgres e fornece uma interface externa para o cluster HAWQ. Todas as interações do usuário passam pelo servidor mestre (Master) que é responsável pela:

* autenticação e autorização;
* análise da consulta;
* geração de planos de consulta, a partir executores de consulta e encaminhamento dos planos.

O servidor mestre (Master) também rastreia o status de execução de consulta e consolida todos os resultados individuais ou intermediários no resultado final.

O mestre é onde o catálogo do sistema global está (GPC - *global system catalog*). O catálogo do sistema global é o conjunto de tabelas do sistema que contêm metadados sobre o próprio sistema HAWQ. O mestre não contém todos os dados do usuário, os dados ficam apenas no HDFS. O mestre autentica conexões de cliente, processa comandos SQL recebidas, distribui a carga de trabalho entre os segmentos, coordena os resultados retornados por cada segmento e apresenta os resultados finais para o programa cliente.

## Servidor em modo de espera (*Standby Master*)

Para lidar com eventuais falhas, o mestre possui um outro servidor correspondente, denominado como o “mestre em modo de espera” (*stand by master*). Ele permanece sincronizado com o mestre através de envio de log. Os metadados são armazenados no nó mestre e são replicados para o *Standby Master* usando WAL. Desta forma, o “mestre em modo de espera” sempre vai estar preparado para assumir o controle quando o mestre não estiver disponível. Se ocorrer shutdown do servidor mestre, o servidor em modo de espera (*Standby Server*) entra no ar, porém perde-se as sessões ativas.

## Segmentos (*Segments*)

Os segmentos são as unidades que processam dados simultaneamente e têm nós de dados HDFS (*DataNodes*). Nós de dados são alocados nos segmentos para melhor disponibilidade de dados. Cada host possui apenas um segmento físico e eles podem disparar diversos executores de consulta (QEs - *Query Executors*) para cada parte ou fatia de consulta. Desta forma, permite que um único segmento atue como se fosse vários segmentos virtuais, permitindo que o HAWQ possa utilizar melhor os recursos disponíveis.

Os segmentos virtuais foram configurados para aumentar a utilização dos recursos da arquitetura. Um segmento virtual se comporta como “uma caixa” (um receptáculo) para executores de consulta (QEs - *Query Executors*). Cada segmento virtual tem um executor de consulta para cada fatia de uma consulta. O número de segmentos virtuais usados ​​determina o grau de paralelismo (DOP - *degree of parallelism*) de uma consulta.

Um **segmento** se difere um **mestre**, por exemplo, pelas seguintes características:

1. É um nó sem estado (*stateless*);
2. não armazena metadados;
3. não armazena dados no sistema de arquivos local.

O mestre envia a solicitação SQL aos segmentos juntamente com as informações de metadados relacionados com o processo. Os metadados contém a url do HDFS para a tabela necessária. O segmento acessa os dados correspondentes usando este URL.

## HAWQ *Network Interconnect* (Interconexão de rede)

A interconexão de rede é a camada de rede de HAWQ. Quando um usuário se conecta a um banco de dados e executa uma consulta, os processos são criados em cada segmento para lidar com a consulta. A interconexão de rede se refere à comunicação entre processos e os segmentos, utilizando padrão *Ethernet switching* para executar essa comunicação.

A interconexão de rede usa UDP (*User Datagram Protocol*) para enviar mensagens através da rede como protocolo padrão de comunicação. O software HAWQ realiza a verificação de pacotes adicionais além do que é fornecido pela UDP. Isto significa que a confiabilidade é equivalente ao protocolo TCP (*Transmission Control Protocol*), contudo o desempenho e a escalabilidade excedem o de TCP. TCP tem um limite de escalabilidade de 1000 instâncias de segmento, porém com o UDP como o protocolo padrão, este limite não é aplicável, permitindo um limite praticamente ilimitado de instâncias de segmento.

## Acesso a sistemas externos / *Pivotal eXtension Framework* (PXF)

PXF (*Pivotal eXtension Framework*) é um framework rápido e extensível que conecta o HAWQ a diferentes mecanismos de armazenamento em HDFS e disponibiliza uma API paralela.

HDFS

HBase

Hive



Xtension Framework



**Figura 3.3:** PxF – *Pivotal eXtension Framework*

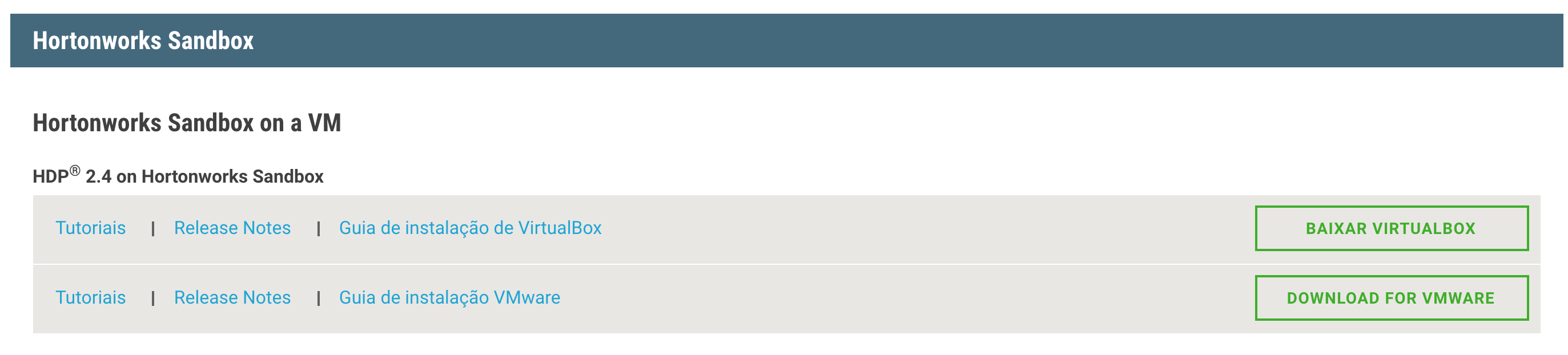
As principais características do PxF são as seguintes:

* Versão avançada de tabelas externas
* Possibilidade de combinar dados do HAWQ e do Hadoop na mesma consulta
* Suporta conectores para HDFS, HBase e Hive
* Disponibiliza um framework extensível que permite o desenvolvimento de novos conectores para qualquer fonte de dados
* Habilidade de ler diferentes tipos de dados do HDFS
  + Arquivos texto, comprimidos e não comprimidos
  + Arquivos sequenciais
  + Arquivos AVRO
* Habilidade de ler dados de fontes externas
  + HBase
  + Cassandra
  + Redis
* API extensível

# **Instalação**

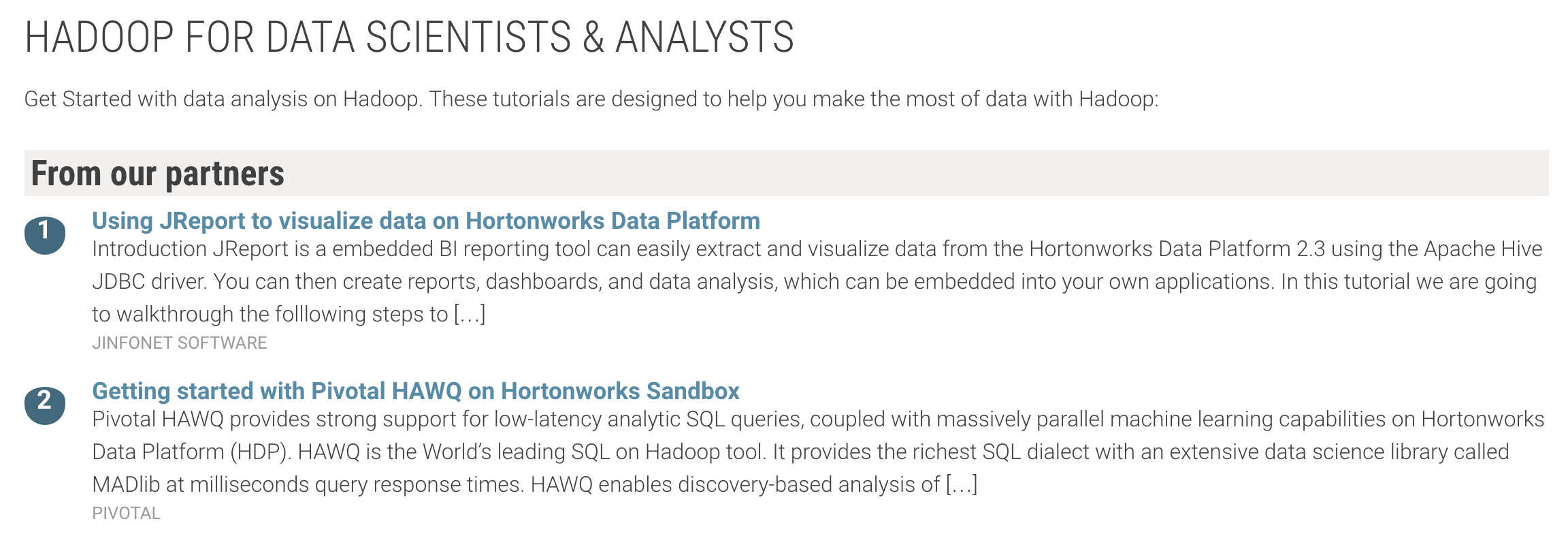
Para o presente estudo foi utilizado uma máquina virtual, onde o HAWQ já estava instalado e configurado. Desta forma, a descrição sobre a instalação não se aplica como objeto do estudo ou escopo do presente estudo.

Foi utilizada uma *sandbox* da empresa Hortonworks, mais precisamente a HDP® 2.4 *on Hortonworks Sandbox*, versão para virtual box que encontra-se no endereço web <http://br.hortonworks.com/downloads/>



**Figura 4.1:** Tela capturada do Site da Web da HDP® 2.4 *on Hortonworks Sandbox*

Neste site web há um tutorial completo sobre os primeiro passos com o HAWQ, na seção *HADOOP FOR DATA SCIENTISTS & ANALYSTS* esub seção *From our partners* (de nossos parceiros) e com o nome *Getting started with Pivotal HAWQ on Hortonworks Sandbox.* Este tutorial encontra-se no endereço web <http://br.hortonworks.com/hadoop-tutorial/getting-started-with-pivotal-hawq-on-hortonworks-sandbox/> ou você pode simplesmente clicar no link ***Tutorials*** na página de download da *sandbox.*



**Figura 4.2:** Tela capturada do Site da Web dos Tutoriais da Hortonworks

A seguir serão apresentados os principais passos para dar os primeiros passos no HAWQ extraídos do tutorial *Getting started with Pivotal* ***HAWQ*** *on Hortonworks Sandbox* para *VirtualBox*. Para mais informações, recomenda-se que se acesse o endereço web que contem os tutoriais.

## Iniciando o HAWQ

Recomenda-se que se configure a **VM** (Máquina virtual) com **4** cores e **12GB** (gigabytes) de memória.

Inicie a **VirtualBox** e efetue *loggin* via terminal no **HDP-HAWQ**. Depois que você se loga na máquina virtual mude para o usuário administrador do **HAWQ** que chamado de “gpadmin“. O **gpadmin** é o super usuário (*power user*) do **HAWQ** e também é o usuário mais amplamente usado pelos administradores.

$ sudo su - gpadmin  
$ source /usr/local/hawq/greenplum\_path.sh

Com este usuário você estará habilitado a executar todos os comandos administrativos do HAWQ localizados sob o diretório bin do HAWQ. Você pode executar qualquer um desses comandos com a opção **-help** para descrever a lista completa de comandos e parâmetros. Para ver uma lista de comandos disponíveis execute os comandos a seguir.

$ ls /usr/local/hawq/bin/  
$ gpstate –help

Para iniciar o HAWQ utilize o comando **gpstart**.

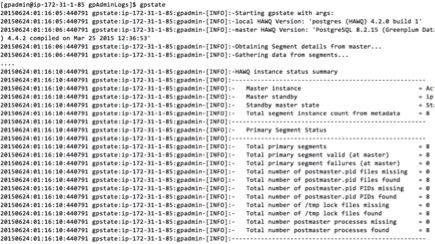
$ gpstart

Também é possível executar o **gpstart** com alguns parâmetros.

$ gpstop -r –a

Após iniciar o serviço, pode-se testar se o **HAWQ** está funcionando corretamente. Para realizar esta checagem, utilize o comando **gpstate**, este comando mostrará um relatório sobre o estado do **HAWQ**, indicando se está ativo e funcionando conforme esperado.

$ gpstate



**Figura 4.3:** Tela capturada após a execução do comando **gpstate.**

Para a lista completa dos comandos administrativos do HAWQ, consulte a página **Utilitário de Gerenciamento de Referência** (*Management Utility Reference*). Você também pode ler sobre variáveis de ambiente, parâmetros de configuração do servidor e conjunto de ferramentas HAWQ.

No mesmo tutorial de onde foi extraído esses primeiros passos, também há passo-a-passo para:

1. CONNECTING TO HAWQ VIA PSQL COMMAND LINE
2. LOAD SAMPLE DATABASE
3. RUNNING QUERIES
4. USING OTHER TOOLS TO WORK WITH HAWQ
5. CONNECTING TO HAWQ VIA PGADMIN

Estes passos adicionais complementam a série de primeiros passos que permitem uso básico e essencial do **HAWQ**.

# **Interação**

A interação com o HAWQ é relativamente simples, principalmente pelo fato de que a linguagem de consulta é SQL ANSI. Desta forma, acelera o processo de aprendizado e uso da ferramenta, pois não exige a necessidade de se aprender uma nova linguagem de consulta.  Nesse sentido, a seguir serão apresentados um conjunto de scripts básicos de SQL.

A seguir serão apresentadas as principais operações utilizando HAWQ, como:

* Inserção;
* Atualização;
* Exclusão;
* Etc.

## SELECT

Este comando serve para consultar dados (selecionar dados de uma os mais tabelas).

Exemplo 1 do comando SELECT:

select \*   
 from retail\_demo.fat\_por\_zip  
 order by total desc limit 10;

Exemplo 2 do comando SELECT:

select billing\_address\_postal\_code,  
 sum(total\_paid\_amount::float8) as total,  
 sum(total\_tax\_amount::float8) as tax  
 from retail\_demo.orders\_pxf  
 group by billing\_address\_postal\_code  
 order by total desc limit 10;

## INSERT

O comando INSERT adiciona um registro na base de dados.

Exemplo do comando INSERT:

insert into retail\_demo.categories\_dim\_hawq values (666, ‘Livros de BigData e Machine Learning’);

## UPDATE

Como o HAWQ é append-only ele não suporta update.

## DELETE

O produto não suporta DELETE atômico, somente limpeza completa da tabela.

TRUNCATE TABLE retail\_demo.orders\_hawq;

## CREATE TABLE

Este comando que usa para criar tabelas no HAWQ.

Exemplo 1 do comando CREATE TABLE:

CREATE TABLE demo.nasa\_weblogs (  
 host STRING,  
 identity STRING,  
 user STRING,  
 time STRING,  
 request STRING,  
 status STRING,  
 size STRING,  
 referer STRING,  
 agent STRING)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'  
WITH SERDEPROPERTIES (  
 "input.regex" = "([^ ]\*) ([^ ]\*) ([^ ]\*) (-|\\[[^\\]]\*\\]) ([^ \"]\*|\"[^\"]\*\") (-|[0-9]\*) (-|[0-9]\*)(?: ([^ \"]\*|\"[^\"]\*\") ([^ \"]\*|\"[^\"]\*\"))?"  
)  
STORED AS TEXTFILE  
LOCATION '/hawq\_demo/weblogs/';

Exemplo 2 do comando CREATE TABLE:

CREATE TABLE retail\_demo.categories\_dim\_hawq (  
 category\_id integer,  
 category\_name character varying(400)   
)  
WITH (appendonly=true, compresstype=quicklz) DISTRIBUTED RANDOMLY;

## DROP TABLE

Exemplo do comando DROP TABLE:

DROP TABLE retail\_demo.orders\_hawq;

# **Tarefas Administrativas**

Neste tópico será apresentada a visão geral das principais tarefas administrativas do HAWQ. Serão abordadas as tarefas de como realizar *back up* (cópia de segurança dos dados), como restaurar (*restoring*) os dados a partir de um *back up* e algumas questões de autenticação.

No dia-a-dia de um administrador de banco de dados é necessário realizar cópias de segurança dos dados (*back up*) e restaurar os dados quando a versão original dos dados estiver com algum problema. O HAWQ possui três ferramentas (*utilities*) para ajudar administradores a fazerem *back up* dos dados:

* **gpfdist**;
* **PXF**;
* **pg\_dump.**

## Diferenças entre gpfdist e PXF

O **gpfdist** e o **PXF** provém o melhor desempenho. São ferramentas que permitem cargas que podem rodar em paralelo (*parallel loading*) e que realizam a descarga (*unloading*) de todos os dados para tabelas externas. Os arquivos de *back up* podem ficar em um sistema de arquivos local ou em HDFS. Para recuperar dados, você pode recarregar dados de volta das tabelas externas ao banco de dados.

Apesar de serem utilitários semelhantes, eles apresentam diversas diferenças. Essas diferenças podem ser definitivas para escolher qual ferramenta utilizar. A seguir, serão apresentadas as principais diferenças entre o **gpfdist** e o **PXF.**

|  |  |
| --- | --- |
| **gpfdist** | **PXF** |
| Mantém os arquivos no sistema de arquivos local (*local file system*) | Mantém os arquivos no HDFS |
| Somente suporta texto simples como formato | Suporta testo, mas também suporta formatos binários como o AVRO e outros formatos semelhantes |
| Não suporta compressão de dados | Suporta compressão de dados, inclusive é possível escolher o ***codec*** para realizar a compressão |
| Ambos apresentam excelente desempenho, porém o **gpfdist** é maisrápido | |

Em resumo, o PXF é mais flexível e apresenta mais opções para sua utilização, enquanto o gpfdist é mais rápido, porém possui menos opções para a sua utilização.

Para a maioria dos casos o **gpfdist** e o **PXF** são excelentes opções para *back up* paralelo (*parallel back up*). Porém, há pelo menos duas situações que requerem atenção, onde talvez não se devesse fazer *back up* paralelo ou operações de restauração de *back up*. Elas são as seguintes:

* Executar *backups* incrementais periodicamente;
* Despejar um grande volume de dados para tabelas externas, pois esse processo tende a levar um longo tempo.

Nestas situações, você pode fazer *backup* de dados durante os processos de ETL e recarregar no HAWQ. A Pivotal recomenda esta abordagem devido à natureza progressiva e a flexibilidade de escolher onde se deseja armazenar os arquivos de *backup* desta abordagem.

## Uso do pg\_dump e pg\_restore

O HAWQ suporta o **pg\_dump** e **pg\_restore** – os utilitários de *back up* e *restore* – do Postgres. O utilitário **pg\_dump** cria um único, grande arquivo de despejo no ***master host,*** contendo todos os dados de cada um dos segmentos ativos (***segments***). O utilitário **pg\_restore** restaura um banco de dados HAWQ do arquivo gerado pelo **pg\_dump**. Porém, a sua utilização não é muito prática, pois dependendo do tamanho dos dados, pode não haver espaço em um ***master host*** para armazenar todos os dados que estão distribuídos nos segmentos em um único arquivo. Algo que também não faz muito sentido, uma vez que vai na direção contrária de um dos motivos pelos quais os banco de dados distribuído foram pensados, poder crescer horizontalmente e armazenar quantidades massivas de dados. Na verdade, o **HAWQ** mantém este utilitário para os casos em que os usuários desejem migrar os dados do **Postgres** para o **HAWQ.**

A seguir os passos para realizar *back up* com **pg\_dump**:

a) Para criar um arquivo *back up*:

$ pg\_dump -Ft -f mydb.tar mydb

b) Para criar um arquivo *back up* com compressão:

$ pg\_dump -Fc -Z3 -f mydb.dump mydb

c) Para restaurar um arquivo de *back up:*

$ pg\_restore -d new\_db mydb.dump

## Estratégias de *back up*

A tabela a seguir resume as estratégias de back up do HAWQ para alguns tipos de situações:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Necessidade** | **gpfdist** | **PXF** | **pg\_dump** | **Raw Data Backup** |
| Parallel | Yes | Yes | No | No |
| Incremental Backup | No | No | No | Yes |
| Backup Location | Local FS | HDFS | Local FS | Local FS, HDFS |
| Format | Text, CSV | Text, CSV, Custom | Text, Tar, Custom | Depends on format of row data |
| Compression | No | Yes | Only support custom format | Optional |
| Scalability | Good | Good | --- | Good |
| Performance | Fast loading, Fast unloading | Fast loading, Normal unloading | --- | Fast (Just file copy) |

**Tabela 6.3:** Estratégias de back up do HAWQ (Fonte: [http://hawq.docs.pivotal.io/](http://hawq.docs.pivotal.io/docs-hawq/topics/BackingUpandRestoringHAWQDatabases.html))

## Estimava de espaço requerido

Antes de iniciar efetivamente a tarefa de realizar o *back up,* é necessário calcular se há espaço suficiente para o armazenamento dos dados a serem mantidos como cópia reserva. O HAWQ possui um *administrative schema* para ajudar os usuários a calcular o espaço necessário. Esse *administrative schema* chama-se **hawq\_toolkit**. Ele pode ser usado para consultar os catálogos do sistema (*system catalogs*), os arquivos de log (*log files*) e o ambiente (*operating environment*) para obter informações de status do sistema. O *administrative schema* **hawq\_toolkit** contém uma série de “comandos” que você pode acessar usando junto com os comandos SQL.

Para calcular o tamanho do banco de dados, chame **hawq\_toolkit** e a opção **hawq\_size\_of\_database** conforme a seguir:

mydb=# SELECT sodddatsize   
 FROM **hawq\_toolkit.hawq\_size\_of\_database**   
 WHERE sodddatname=’mydb’;

**Ponto de atenção:** para as tabelas pesquisadas no banco de dados que estiverem compactadas, o tamanho apresentado será o tamanho que possuem quando compactadas.

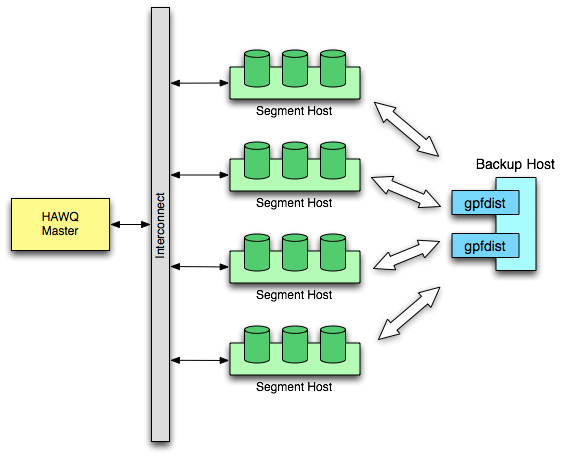
A seguir, alguns pontos para se levar em consideração quando os cálculos de volume forem realizados:

* Se as tabelas do banco de dados e dos arquivos de backup estão compactados, você pode usar o valor do **soddatsize**como o valor estimado;
* Se as tabelas de banco de dados estão compactados e os arquivos de *back up* não estão, é necessário multiplicar o valor do **soddatsize**pela taxa de compressão (*compression ratio*);
* Se os arquivos de *back up* estão compactados e as tabelas do banco de dados não estão, é necessário dividir **soddatsize**pela taxa de compressão (*compression ratio*);
* Se está sendo utilizando **HDFS** com **PXF**, o requisito de espaço é a formula: **size\_of\_backup\_files \* replication\_factor**;
* Se está sendo usado o **gpfdist**, a necessidade de espaço para cada instância **gpfdist** é a seguinte fórmula, desde os dados da tabela estejam distribuídos uniformemente em cada uma das instâncias do **gpfdist** **size\_of\_backup\_files / num\_gpfdist\_instances**;

## Utilitário gpfdist

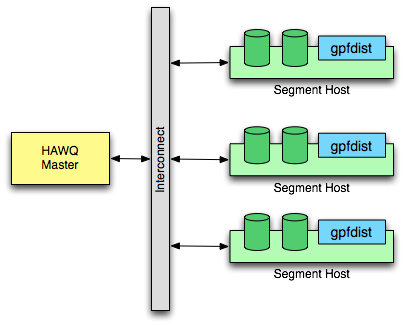
O **gpfdist** é um dos programas que realiza a distribuição paralela de dados do HAWQ. Para usar o **gpfdist**, inicie o programa **servidor do gpfdist**, no ***host*** onde se deseja armazenar os arquivos de *back up*. É possível iniciar múltiplas instâncias de **gpfdist**, no mesmo ou em diferentes *hosts.* Para cada instância de **gpfdist**, deve-se especificar um diretório. Por exemplo, se for uma máquina dedicada com dois discos, é possível iniciar duas instâncias e cada uma poderia usar um disco.

A imagem a seguir apresenta um ambiente com múltiplas instâncias de **gpfdist** em um *host* de *back up.*



## Figura 6.5.1: ambiente com múltiplas instâncias de gpfdist.

A imagem a seguir representa um ambiente com um instância **gpfdist** em cada segmento de *host* (*segment host).*



## Figura 6.5.2: ambiente com um instância gpfdist em cada *segment host*.

## Usando gpfdist para fazer back up

A seguir um conjunto de passos para executar fazer o back up de uma base de dados tpch.

**Passo 1:** Crie os caminhos de back up e inicie as instâncias gpfdist.

$ mkdir -p /data1/gpadmin/backup/tpch\_20140627 /data2/gpadmin/backup/tpch\_20140627  
$ gpfdist -d /data1/gpadmin/backup/tpch\_20140627 -p 8080 &  
$ gpfdist -d /data2/gpadmin/backup/tpch\_20140627 -p 8081 &

Comentários do código acima:

* O primeiro comando cria dois diretórios em dois discos com o mesmo sufixo “backup/tpch\_20140627”;
* Estes dois diretórios são carregados como os caminhos de *back up* da base de dados **tcph** em 2014-06-27.
* Nos dois comando subsequentes, o exemplo mostra duas instâncias de gpfdist, uma usando a porta 8080 e a outra usando a porta 8081.

**Passo 2**: Salve o esquema (*schema*) da base de dados.

master\_host$ pg\_dump --schema-only -f tpch.schema tpch  
master\_host$ scp tpch.schema sdw1:/data1/gpadmin/backup/tpch\_20140627

Comentários do comando acima:

* No *master host* HAWQ o utilitário **pg\_dump** foi usado para salvar o *schema* da base de dados tcph no arquivo chamando “tpch.schema”
* Copie esse arquivo para caminho do backup quando quiser restaurar o *database schema.*

**Passo 3:** Salve o esquema (schema) da base de dados.

master\_host$ psql tpch  
tpch=# create writable external table wext\_orders (like orders) location ('gpfdist://sdw1:8080/orders1.csv', 'gpfdist://sdw1:8081/orders2.csv') format 'CSV';  
tpch=# create writable external table wext\_lineitem (like lineitem) location ('gpfdist://sdw1:8080/lineitem1.csv', 'gpfdist://sdw1:8081/lineitem2.csv') format 'CSV';

Comentários do comando acima:

* O exemplo mostra duas tabelas no banco de dados **tpch**: “orders” e “line item” .
* O exemplo mostra que duas tabelas externas correspondentes foram criadas para elas.
* É necessário especificar um caminho ou uma instância de **gpfdist** na cláusula de **LOCATION**.
* Este exemplo usa o formato de texto CSV, mas poderiam ser usados outros formatos.

**Passo 4:** Descarregue os dados nas tabelas externas (*external tables*).

tpch=# begin;  
tpch=# insert into wext\_orders select \* from orders;  
tpch=# insert into wext\_lineitem select \* from lineitem;  
tpch=# commit;

**Passo 5:** (Opcional) Parar os servidores gpfdist para liberar as portas para outros processos.

$ ps -ef | grep gpfdist  
$ kill 612368; kill 612369

Comentários do comando acima:

* Procure o **progressId** e mate o processo;
* 612368 e 612369 são os números de progressId do exemplo acima.

## Restaurando um back up com gpfdist

A seguir um conjunto de passos para restaurar um back up de uma base de dados utilizando **gpfdist**.

**Passo 1:** Reinicie as instâncias **gpfdist** se elas não estiverem rodando.

sdw1$ gpfdist -d /data1/gpadmin/backup/tpch\_20140627 -p 8080 &  
sdw1$ gpfdist -d /data2/gpadmin/backup/tpch\_20140627 -p 8081 &

**Passo 2:** Crie uma nova base de dados e restaure o *schema.*

master\_host$ createdb tpch2  
master\_host$ scp sdw1:/data1/gpadmin/backup/tpch\_20140627/tpch.schema .  
master\_host$ psql -f tpch.schema -d tpch2

**Passo 3:** Crie um tabela externa (*readable external table*) para cada tabela da base de dados.

tpch2=# create external table rext\_orders (like orders) location ('gpfdist://sdw1:8080/orders1.csv', 'gpfdist://sdw1:8081/orders2.csv') format 'CSV';  
tpch2=# create external table rext\_lineitem (like lineitem) location ('gpfdist://sdw1:8080/lineitem1.csv', 'gpfdist://sdw1:8081/lineitem2.csv') format 'CSV';

**Passo 4:** Recarregue os dados a partir das tabelas externas

tpch2=# insert into orders select \* from rext\_orders;  
tpch2=# insert into lineitem select \* from rext\_lineitem;

**Passo 5:** Execute o comando **ANALYZE**, após ter carregado todos os dados.

tpch2=# analyze;

## Usando PXF para fazer back up

O **PXF** (*Pivotal eXtension Framework*) é um framework que ao **HAWQ** a interagir por SQL query com sistemas externos. A seguir serão apresentados o conjunto de passos para se realizar *back up*.

**Passo 1:** crie um folder no **HDFS** para realizar o *back up*.

$ hdfs dfs -mkdir -p /backup/tpch-2014-06-27

**Passo 2:** esvazie o *database schema* usando **pg\_dump** e armazene o arquivo de *schema* na pasta de *back up.*

$ pg\_dump --schema-only -f tpch.schema tpch  
$ hdfs dfs -copyFromLocal tpch.schema /backup/tpch-2014-06-27

**Passo 3:** crie uma tabela externa com permissão de escrita (*writable external table*) para cada tabela do base de dados.

$ psql tpch  
  
tpch=# create writable external table wext\_orders (like orders) location ('pxf://namenode\_host:51200/backup/tpch-2014-06-27/orders?Profile=  
HdfsTextSimple&COMPRESSION\_CODEC=org.apache.hadoop.io.compress.SnappyCodec') format 'TEXT';  
  
tpch=# create writable external table wext\_lineitem (like lineitem) location ('pxf://namenode\_host:51200/backup/tpch-2014-06-27/lineitem?Profile=  
HdfsTextSimple&COMPRESSION\_CODEC=org.apache.hadoop.io.compress.SnappyCodec') format 'TEXT';

**Passo 4:** descarregue os dados para a tabela externa (*external table*)

tpch=# begin;  
tpch=# insert into wext\_orders select \* from orders;  
tpch=# insert into wext\_lineitem select \* from lineitem;  
tpch=# commit;

**Passo 5:** (opcional) mude o fator de replicação do arquivo **HDFS** para a pasta de *back up*. O **HDFS** replicará cada bloco em três blocos como comportamento padrão (*default of replication factor*) por questões de confiabilidade. É possível diminuir esse número para o arquivo de *back up* se necessário.

master\_host$ hdfs dfs -setrep 2 /backup/tpch-2014-06-27

Comentário: Isso apenas se aplica para o fator de replicação dos arquivos existentes. Novos arquivos continuarão usando o padrão para o fator de replicação.

## Restaurando um back up com PXF

A seguir serão apresentados o conjunto de passos para se restaurar um *back up* no HAWQ com PXF.

**Passo 1:** crie uma nova base de dados e restaure o *schema.*

$ createdb tpch2  
$ hdfs dfs -copyToLocal /backup/tpch-2014-06-27/tpch.schema .  
$ psql -f tpch.schema -d tpch2

**Passo 2:** crie uma tabela externa para leitura (*readable external table*) para cada tabela que deverá ser restaurada.

$ psql tpch2  
tpch2=# create external table rext\_orders (like orders) location ('pxf://namenode\_host:51200/backup/tpch-2014-06-27/orders?Profile=HdfsTextSimple') format 'TEXT';  
tpch2=# create external table rext\_lineitem (like lineitem) location ('pxf://namenode\_host:51200/backup/tpch-2014-06-27/lineitem?Profile=HdfsTextSimple') format 'TEXT';

**Passo 3:** carregue os dados de volta a partir das tabelas externas.

tpch2=# insert into orders select \* from rext\_orders;  
tpch2=# insert into lineitem select \* from rext\_lineitem;

**Passo 4:** execute o comando ANALYZE após ter carregado os dados.

tpch2=# analyze;

## Passos básicos de back up e restauração

Nesta seção, serão apresentados os passos básicos de *back up* e restauração no HAWQ, independentemente do utilitário a ser atualizado. Os passos a seguir não demonstrarão comandos, apenas serão um conjunto de passos genéricos para guiar sobre as principais atividades de *back up* e restauração.

**A) Como fazer cópia de segurança (Back up)**

A seguir um conjunto de passos para executar um *parallel backup*:

1. Verifique o tamanho do banco de dados para garantir que o sistema de arquivos possua o espaço suficiente para salvar os arquivos de *back up*.
2. Use o utilitário **pg\_dump** para gerar o esquema (*schema file*) do banco de dados *back up* (destino).
3. Crie uma tabela externa (que permita atualizações) para cada tabela do banco de dados que se deseje realizar o *back up*.
4. Copie os dados das tabelas origem para as tabelas destino criados no banco de dados (destino).

**B) Como restaurar cópia de segurança (restoring)**

A seguir um conjunto de passos para recuperar os dados a partir de um *back up*:

1. Crie um banco de dados onde se deseja recuperar os dados.
2. Recrie o esquema do arquivo de ***schema*** (criado durante o processo **pg\_dump**).
3. Criar uma tabela externa legível para cada tabela no banco de dados.
4. Carregar dados da tabela externa para a tabela real.
5. Execute o comando **ANALYZE** após o carregamento estiver completo. Isso garante que o planejador de consultas gere um plano ideal com base nas estatísticas.

# **Evidências**

Nos próximos tópicos serão apresentadas as evidência das execuções dos scripts SQL descritos no tópico 5. Interações que descreve os principais commandos e seus usos.

## SELECT

A seguir a evidência do comando SELECT:

[gpadmin@sandbox ~]$ psql

psql (8.2.15)

Type "help" for help.

gpadmin=# select \*

gpadmin=# select \*

gpadmin-# from retail\_demo.fat\_por\_zip

gpadmin-# order by total desc limit 10;

billing\_address\_postal\_code | total | tax

-----------------------------+-----------+-----------

48001 | 111868.32 | 6712.0992

15329 | 107958.24 | 6477.4944

42714 | 103244.58 | 6194.6748

41030 | 101365.5 | 6081.93

50223 | 100511.64 | 6030.6984

03106 | 83566.41 | 0

57104 | 77383.63 | 3095.3452

23002 | 73673.66 | 3683.683

25703 | 68282.12 | 4096.9272

26178 | 66836.4 | 4010.184

(10 rows)

gpadmin=# select billing\_address\_postal\_code,

gpadmin-# sum(total\_paid\_amount::float8) as total,

gpadmin-# sum(total\_tax\_amount::float8) as tax

gpadmin-# from retail\_demo.orders\_pxf

gpadmin-# group by billing\_address\_postal\_code

gpadmin-# order by total desc limit 10;

billing\_address\_postal\_code | total | tax

-----------------------------+-----------+-----------

48001 | 111868.32 | 6712.0992

15329 | 107958.24 | 6477.4944

42714 | 103244.58 | 6194.6748

41030 | 101365.5 | 6081.93

50223 | 100511.64 | 6030.6984

03106 | 83566.41 | 0

57104 | 77383.63 | 3095.3452

23002 | 73673.66 | 3683.683

25703 | 68282.12 | 4096.9272

26178 | 66836.4 | 4010.184

(10 rows)

## INSERT

A seguir a evidência do comando INSERT:

gpadmin=# insert into retail\_demo.categories\_dim\_hawq values ( 666, 'Livros de BigData e Machine Learning' );

INSERT 0 1

gpadmin=# select \* from retail\_demo.categories\_dim\_hawq where category\_id = 666;

category\_id | category\_name

-------------+--------------------------------------

666 | Livros de BigData e Machine Learning

(1 row)

## DELETE (TRUNCATE)

A seguir a evidência do comando DELETE:

gpadmin=# TRUNCATE TABLE retail\_demo.orders\_hawq;

TRUNCATE TABLE

gpadmin=# select count(\*) from retail\_demo.orders\_hawq;

count

-------

0

(1 row)

## CREATE TABLE

A seguir a evidência do segundo comando CREATE TABLE:

gpadmin=# \d retail\_demo.categories\_dim\_hawq

Append-Only Table "retail\_demo.categories\_dim\_hawq"

Column | Type | Modifiers

---------------+------------------------+-----------

category\_id | integer | not null

category\_name | character varying(400) | not null

Compression Type: quicklz

Compression Level: 1

Block Size: 32768

Checksum: f

Distributed randomly

## DROP TABLE

A seguir a evidência do comando SELECT:

gpadmin=# DROP TABLE retail\_demo.orders\_hawq;

DROP TABLE

gpadmin=# \d retail\_demo.orders\_hawq

Did not find any relation named "retail\_demo.orders\_hawq".

# **Conclusões**

Em resumo, recomenda-se o uso do HAWQ e baseado nos estudos contidos neste presente trabalho acredita-se que apesar de relativamente novo, é uma excelente aposta como um banco de dados noSQL para BigData. Permite integrações com HDFS e com outras bases de dados. O fato de a linguagem de consulta ser SQL, ajuda na curva de aprendizagem. Adicionalmente, a questão de ter sido projetado para alto desempenho em quantidades massivas de dados, ele apresentou em vários testes um alto desempenho e bate os concorrentes neste sentido. Outro fator que contribui com o uso do HAWQ é a integração com MADLib que permite acesso direto a uma biblioteca de *machine learning*, habilitando o uso de *machine learning* praticamente de forma nativa.

A seguir, serão apresentados os principais motivos com mais detalhes para reforçar a recomendação de uso do HAWQ comentada no parágrafo anterior.

## Quadro comparativo

Em um estudo comparativo entre as ferramentas com maior semelhança com o HAWQ ele é a melhor opção na maioria dos quesitos. O quadro seguir apresenta uma visão geral desta comparação entre as ferramentas.

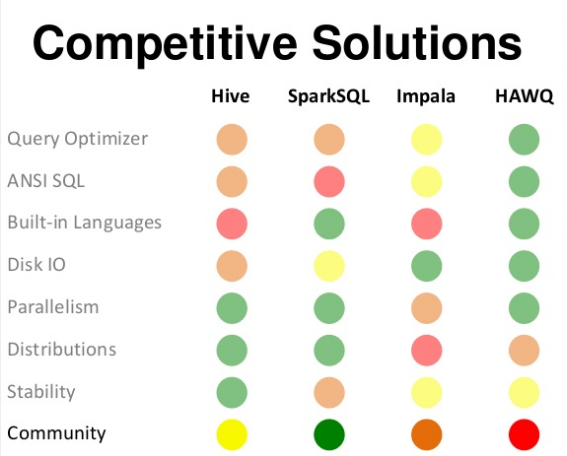


Figura 8.1: Comparação HAWQ com outras ferramentas.

Nos quesitos otimizador de *query* (*query optimizer*) e SQL ANSI (*ANSI SQL*) o HAWQ é o melhor entre todas as ferramentas comparadas. Inclusive, na questão SQL ANSI, em um outro estudo comparativo com o Impala, o HAWQ rodou 100% das *queries* SQL ANSI enquanto o Impala apresentou compatibilidade parcial, rodando 55 das 99 queries SQL ANSI sem modificações. Outros pontos a serem destacados sobre o HAWQ neste quesito:

* 100% compatível com TPC-DS
* HAWQ completou 58/99 consultas TPC-DS 12 horas antes em relação ao **Impala**;
* Consultas multi-dimensionais com subqueries, eliminação dinâmica de partições, *joins* de tabelas grandes, e roll-ups;
* Permite alta concorrência.

Nos quesitos **distribuições** (*distributions*), **estabilidade** (*stability*) e **comunidade** (*community*) são o HAWQ é batido pelos outros competidores. Porém, como se trata de uma ferramenta relativamente nova, está em estágio de consolidação e desenvolvimento comunidade. Existe a tendência de que num curto espaço de tempo estas questões de estabilidade e comunidade sejam resolvidas, pois os adeptos ao HAWQ estão crescendo.

Nas questões de desempenho e processamentos massivo de dados o HAWQ bate todos os competidores. Lembrando que o HAWQ foi projetado para permitir processamento massivo de dados e promover um alto desempenho na velocidade de resposta de consulta.

O seu otimizador de query é o mais rápido, em relação aos competidores comparados. Ele pode utilizar o otimizador do Postgres ou o ORCA. O ORCA é o otimizador de query desenvolvido especialmente para o HAWQ, justamente para habilitar alto desempenho em volumes massivos de dados. O otimizador se baseia em custo e se prepara através de um plano robusto de execução. Além de estar apto para o gerenciamento de dados complexos. Os principais benefícios do otimizador são:

* Otimiza performance e custos
* Maximiza recursos do cluster HDP
* Substitui EDW sem comprometer o desempenho

## Integração

Com PXF (Pivotal eXtension Framework) o HAWQ pode se conectar a diferentes mecanismos de armazenamento em HDFS e outros bancos de dados. Também pode se conectar a base de dados via ODBC e JDBC. Neste sentido, além do HDFS ele também pode se conectar diretamente com outras bases de dados o que aumenta a sua abrangência de uso.

Com o Ambari, o gerenciamento dos serviços do HAWQ em Hadoop ficam relativamente fáceis de serem gerenciados. Desta forma, as o HAWQ apresenta os seguintes benefícios:

* Flexibilidade;
* Acessibilidade;
* Portabilidade.

## Curva de aprendizagem

Como o HAWQ foi projetado para usar SQL como linguagem de consulta de dados, não exige a necessidade de se aprender uma nova linguagem de consulta. Quem já conhece SQL, praticamente já está apto para criar queries e fazer consultas no HAWQ. Essa característica, também poderia facilitar a escolha da ferramenta frente outras opções, pois como ponto positivo reduziria a curva de aprendizagem e eventuais questões de desenvolvimento de pessoas.

## MADLib

Com a MADLib o HAWQ consegue fazer aprendizado de máquina avançado para Big Data e realizar operações local, com **operação** diretamente **no banco de dados e de alto desempenho** com MPP/paralela de forma excepcional.

Neste sentido, apresenta os seguintes benefícios:

* Altamente escalável;
* Aprendizado de máquina;
* Acesso direto aos dados no HDP.

# **Referência Bibliográfica**

**[CHANG 2014]** CHANG , LEI et al: **HAWQ: a massively parallel processing SQL engine in hadoop. SIGMOD Conference** 2014: 1223-1234

[**SOLIMAN 2014]** SOLIMAN, MOHAMED A. et al: **Orca: a modular query optimizer architecture for big data. SIGMOD Conference** 2014: 337-348

**[ANTOVA 2014]** ANTOVA, LYBLENA et al: **Optimizing queries over partitioned tables in MPP systems. SIGMOD Conference** 2014: 373-38**4**

**[EL-HELV 2015]** EL-HELV, AMRet al: **Optimization of Common Table Expressions in MPP Database Systems.** PVLDB 8(12): 1704-1715 (2015)

**[HORTONWORKS 01]** HORTONWORKS: **Getting started with Pivotal HAWQ on Hortonworks Sandbox.** Disponível em: <http://hortonworks.com/hadoop-tutorial/getting-started-with-pivotal-hawq-on-hortonworks-sandbox/> Acessado em: 30/05/2016 às 22:30.

**[PIVOTAL 01]** PIVOTAL. **Backing Up and Restoring HAWQ Databases.** Disponível em: [**http://pivotalhd-210.docs.pivotal.io/doc/2100/webhelp/hawq-topics/BackingUpandRestoringHAWQDatabases.html**](http://pivotalhd-210.docs.pivotal.io/doc/2100/webhelp/hawq-topics/BackingUpandRestoringHAWQDatabases.html)Acessado em: 25/06/2016 às 21:30.

**[PIVOTAL 02]** PIVOTAL, **Pivotal HDB 2.0 Release Notes.** Disponível em: <http://hdb.docs.pivotal.io/hdb/releasenotes/HAWQ20ReleaseNotes.html>

**[PIVOTAL 03]** PIVOTAL, **HAWQ Architecture.** Disponível em: <http://hdb.docs.pivotal.io/20/overview/HAWQArchitecture.html> Acessado em: 30/06/2016 às 21:30.

**[HORTONWORKS 02]** HORTONWORKS, **HDP® 2.4 on Hortonworks Sandbox.** Disponível em: <http://br.hortonworks.com/downloads/>

**[HORTONWORKS03]** HORTONWORKS, **HORTONWORKS SANDBOX** Disponível em: <http://hortonworks.com/products/sandbox/#install> Acessado em: 30/05/2016 às 21:30.