

0.0 - Apresentacao	2
1.0 - Introducao e contexto NoSQLBigData	6
1.1.1 - Texto A revolução do Big data	34
1.1.2 - Texto A revolução do Big data Exercicio	40
2.1 - Paradigmas de computação e bancos de dados	41
2.1.1 - Exercício Map Reduce - Palavras	60
2.1.2 - Exercício Map Reduce - SQL	62
2.1.3 - Trabalho Map Reduce	65
2.2 - Paradigmas de computação e bancos de dados	66
3.0 - Bancos de dados chave valor	70
4.0 - Bancos de dados documentos	94
4.1 - Exercício CouchDB	107
5.0 - Bancos de dados colunares	113
5.1 - Exercício Cassandra	132
6.0 - Hadoop	135
6.1 - Exercicio hadoop	163
6.2 - Execicio Pratico Hive	165

NoSQL MongoDB MapReduce Cassandra Python Processamento BerkeleyDB Paralelismo DFS chave-Valor Documentos Colunar

Bancos de dados Distribuídos na Web

Cláudio Lúcio

1

Bancos de dados Distribuídos na Web

Apresentação



Cláudio Lúcio



- Mestre em informática PUC MG, Especialista em estatística UFMG, pós-graduado em gerência de projetos, Bacharel em ciência da computação;
- 18 anos de experiência com Bancos de dados atuando em projetos para clientes do cenário nacional: Arcelor Mittal Tubarão, Banco Mercantil do Brasil, BDMG, BMG, CEMIG, EDS, FIAT, GM do Brasil, Mendes Júnior, Localiza, SEBRAE-SC, SUDECAP, Telefônica, Oi, Vale, VIVO, VMM – Votorantim Mineração e Metais.
- Treinamentos ministrados para várias empresas: Assurant, Athos Pharma, Banco do Brasil, Best Forecast, BM&F, Caixa Econômica, CEMIG, E-Lucid, GM do Brasil, HDI Seguros, Mapfre, Marítima Seguros, Telemar, Telemig Celular e Unibanco ;

Bancos de dados Distribuídos na Web

The screenshot shows the homepage of nosql-database.org. At the top, there's a navigation bar with icons for back, forward, search, and other browser functions. Below the header, the main content area features the "NoSQL" logo on the left, followed by a sub-header: "Your Ultimate Guide to the Non-Relational Universe!". To the right, there's a news feed section with a link to a historic archive from 2009-2011. A horizontal line separates this from the "NoSQL RELATED EVENTS:" section, which lists two upcoming events: "12/14 Sept Flink Forward" and "26 Sept Big Data Guide Event Frankfurt". Further down, there's a section for "Flink Forward" with a small icon of a person at a computer, and a link to register for free. On the far left, under "Wide Column Store / Column Families", there's a link to "Hadoop / HBase API". On the right, there's a section for "NoSQL ARCHIVE" featuring the ArangoDB logo and a globe icon.

Cláudio Lúcio

3

Bancos de dados Distribuídos na Web

Agenda

- Introdução e contexto NoSQL/BigData
- Paradigmas de computação e bancos de dados
 - Demonstração e exercício usando Mincemeat (MapReduce e DFS)
- Bancos de dados baseados em chave valor
- Banco de dados baseado em documentos
 - Demonstração usando CouchDB
- Bancos de dados baseados em colunas
 - Demonstração usando Cassandra
- Hadoop e Hive
- Seminários

Cláudio Lúcio

0.0 - Apresentacao

4

3

Bancos de dados Distribuídos na Web

Avaliações e Frequência

- Chamadas por aula;
- Avaliações e trabalhos:
 - Exercícios:
Leitura e resumo de material sobre Introdução e contexto NoSQL/BigData – 10 pontos;
 - Paradigmas de computação e bancos de dados -30 pontos;
 - Participação nos exercícios práticos – 10 pontos;
 - Seminário – 50 pontos;

Cláudio Lúcio

5

Bancos de dados Distribuídos na Web

Avaliações e Frequência

- Seminário:
 - Grupos com 4 componentes;
 - Temas: MongoDB, HBASE/Hive, REDIS, Voldemort, Neo4J, ArangoDB, Vertica DB, Oracle Berkeley DB, Oracle NoSQL Database, Azure Table Storage, outros.....(validar com o professor)
 - Com base no tema fazer uma apresentação com, por exemplo: Introdução, Características da solução(escalabilidade, consistência, tolerância a partição e modelos de dados), Exemplos de usos práticos e Conclusão
 - Pontos da apresentação:
 - Não é necessário que todos os membros do grupo falem durante a apresentação – mas não é proibido;
 - Observar o tempo da apresentação (atentar ao escopo da apresentação);
 - Data da apresentação: última aula da disciplina (olhar calendário);
 - Tempo de cada apresentação: 15 minutos com no máximo 12 slides (o tempo de apresentação é um dos critérios de avaliação do grupo)
 - O grupo deverá entregar a apresentação por e-mail para o professor;

Cláudio Lúcio

Bancos de dados Distribuídos na Web

Referências Bibliográficas

Brewer, Eric A.: Towards Robust Distributed Systems. Portland, Oregon, July 2000. – ACM Symposium on Principles of Distributed Computing (PODC) on 2000-07-19. Disponível em: <http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf>

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. "Bigtable: A Distributed Storage System for Structured Data"; OSDI'06:2006.

Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters"; pub. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December 2004.

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google File System"; pub. 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October 2003.

Strauch, Christof. NoSQL Databases. Disponível em: <http://www.christof-strauch.de/nosqldb.pdf>. Acesso em: 22/09/2012.

Stonebraker, Michael ; Madden, Samuel ; Abadi, Daniel J. ; Harizopoulos, Stavros; Hachem, Nabil ; Helland, Pat: The end of an architectural era: (it's time for a completerewrite). In: VLDB '07: Proceedings of the 33rd international conference on Very large databases, VLDB Endowment, 2007, p. 1150–1160

Tiwari, Shashank. PROFESSIONAL NoSQL. John Wiley & Sons, Inc. ISBN:978-0-470-94224-6. 2011.

Ullman ,J.D.; Rajaraman ,A. Mining Massive Datasets. Cambridge University Press, UK 2012. Disponível em:
<http://i.stanford.edu/~ullman/mmds.html> . Acesso em: 22/09/2012. Capítulo 2

Vieira, Marcos R; Figueiredo, Josiel M.; Liberatti, Gustavo ; Viebrantz, Alvaro F. M.; Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data. SBBB 2012. Disponível em:

http://data.ime.usp.br/sbbd2012/artigos/pdfs/sbbd_min_01.pdf . Acesso em: 24/10/2012.

Cláudio Lúcio

7

NoSQL MongoDB MapReduce Cassandra Python Processamento
BerkeleyDB Paralelismo DFS
chave-Valor Documentos Colunar

Introdução e contexto NoSQL/Big Data

Cláudio Lúcio

1

Introdução e contexto NoSQL/Big Data

Agenda

- Introdução e conceitos
- Motivações
- Histórico
- Conceitos e Fatos
- Abordagens



Introdução e contexto NoSQL/Big Data

Introdução e conceitos

Cláudio Lúcio

3

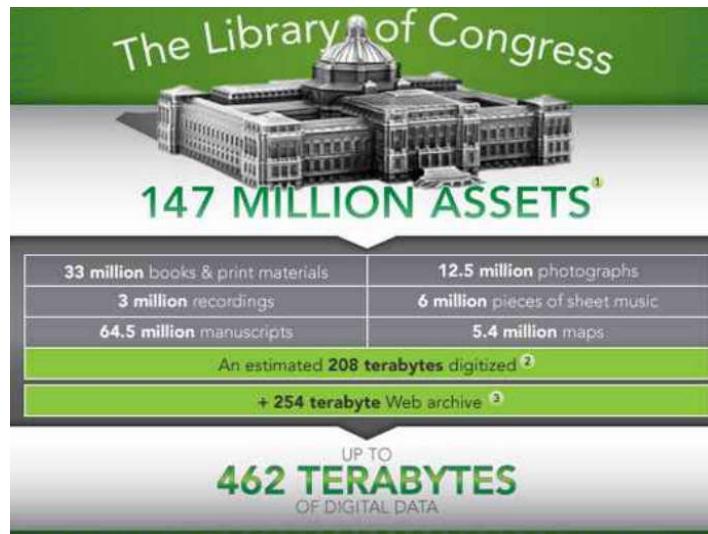
Introdução e conceitos

Termo *NoSQL*:

- Duas palavras: “*No*” e “*SQL*”;
- NonRel: “*No Relational*”;
- *Not Only SQL (RDBMS)*;
- Mais pragmaticamente:
 - Um termo geral para tecnologias de bancos de dados que não utilizam os princípios relacionais;
 - Uso em grandes conjuntos de dados: maior que bilhões de linhas;
 - É toda um classe de produtos e tecnologias para lidar com o paradigma de dados da Web: *Big data*;

Introdução e conceitos

BigData:

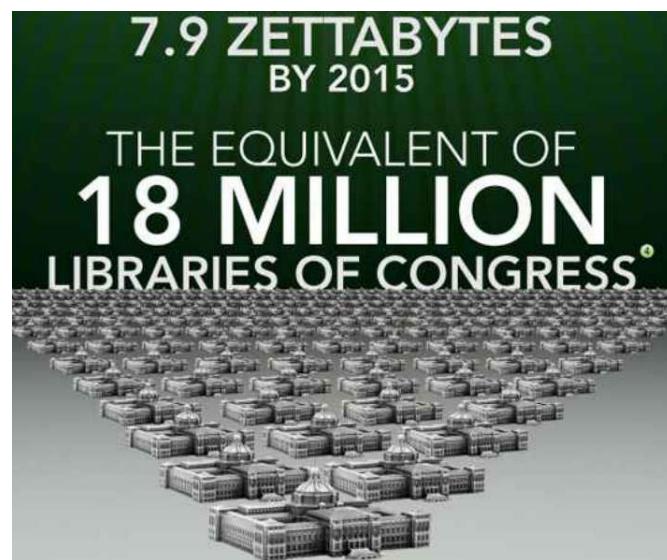


Cláudio Lúcio

5

Introdução e conceitos

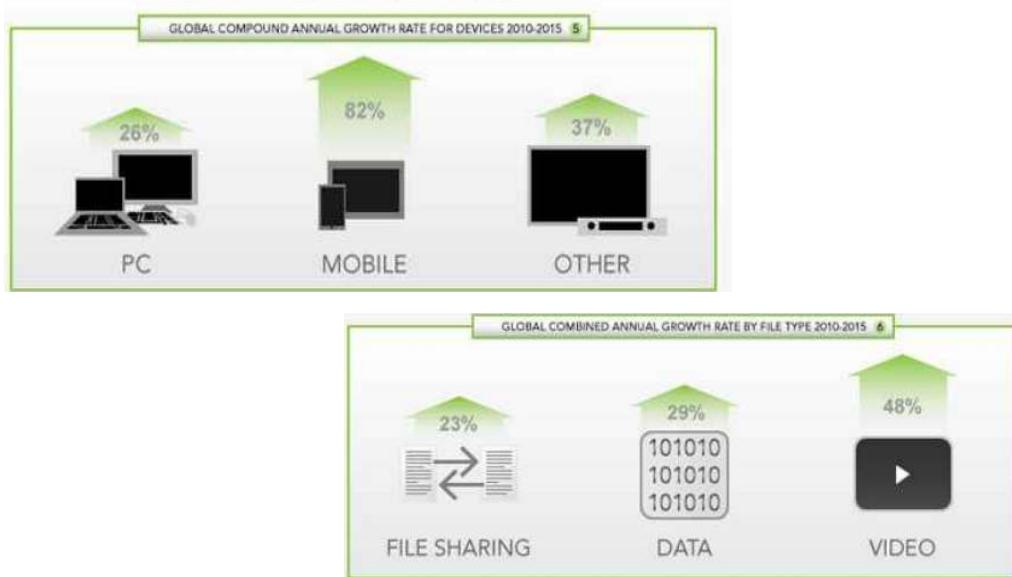
BigData:



Cláudio Lúcio

Introdução e conceitos

BigData:



Cláudio Lúcio

7

Introdução e conceitos

BigData:



- Pesquisa da IDC/EMC apontam um volume de dados na web em 2020 de 35 Zettabytes;
- 3 ou 4 V's: **Variedade, Velocidade, Volume**

+

Valor para os negócios

Kilobyte (kB) — 10^3
Megabyte (MB) — 10^6
Gigabyte (GB) — 10^9
Terabyte (TB) — 10^{12}
Petabyte (PB) — 10^{15}
Exabyte (EB) — 10^{18}
Zettabyte (ZB) — 10^{21}
Yottabyte (YB) — 10^{24}

Introdução e conceitos

Big Data, valor agregado para os negócios:

- Uma rede de supermercados manterá todo o histórico de compras de clientes por produtos, assim como sua rota (RFID) de compra nas lojas;
- Uma rede de locadoras de carro irá reter dados do GPS existente em seus carros. A ideia é entender como os clientes utilizam os carros e oferecer pacotes de descontos de acordo com o uso;
- O Tribunal de Justiça do estado deseja estruturar todos os seus processos, permitindo buscas por advogado, juízes, relatores, redatores, palavras chaves, tipo de causa e outros;

Cláudio Lúcio

9

Introdução e conceitos

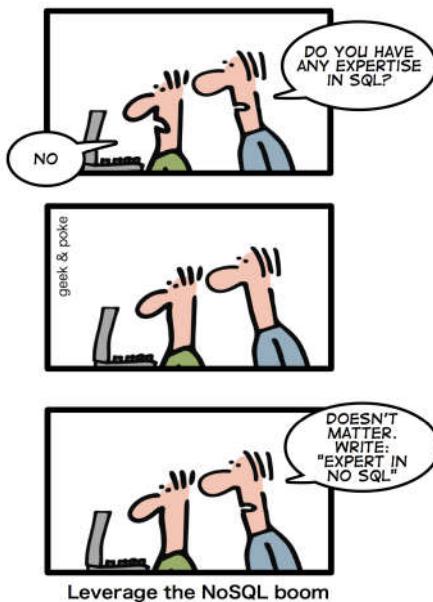
Big Data - Desafios:

- Processamento de volume de dados em milissegundos;
- Armazenar e acessar grandes quantidades de dados. Adicionalmente: tolerância a falhas e política de backups aceitáveis;
- Manipulação eficiente de grandes volumes de dados envolve processamento paralelo e recuperação de falhas em curto espaço de tempo;
- Gerenciamento e manutenção de metadados para dados semi-estruturados e não estruturados gerados de forma contínua por diversos tipos de fontes;

Introdução e conceitos

HOW TO WRITE A CV

Movimento NoSQL:



Cláudio Lúcio

11

Introdução e conceitos

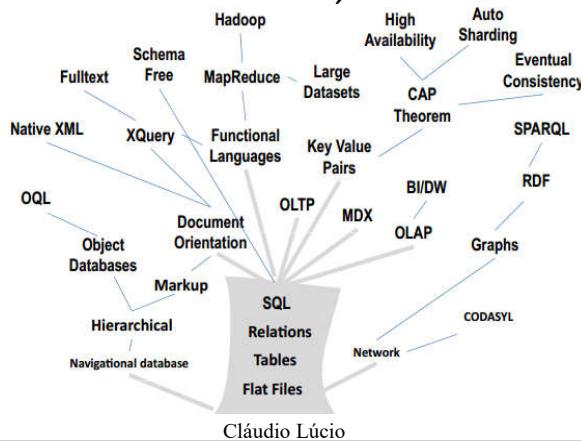
Movimento NoSQL:

- Não é a panacéia: “One size fits all”;
- É mais adequada para *BigData*:
 - Alto desempenho;
 - Escalabilidade para web;
 - Análise de grande volume de dados;
- Aderente a computação nas nuvens:
 - Escalabilidade ao custo acessível a medida que o volume de dados aumenta;
 - Soluções NoSQL são oferecidas como serviços web ;

Introdução e conceitos

NoSQL - Definição:

- *Not Only SQL (not only RDBMS);*
- Um conjunto de produtos e tecnologias para lidar com o paradigma de dados da Web;



13



Introdução e contexto NoSQL/Big Data

Motivações

Motivações

Movimento *NoSQL* - Motivações:

- Elevado *Throughput*
 - A solução *Bigtable(Google)* permite buscas locais em 1 bilhão de celulas de dados;
 - *Bigtable(Google)* processa 20 *petabytes* por dia utilizando a abordagem *map reduce*;
- Escalabilidade Horizontal utilizando hardware de baixo custo
 - Custo de uma solução *Oracle Real Application Clusters (RAC)* com *Automatic Storage Management (ASM)*;
 - Lista de preço 04/09/2012:

Oracle Database				
Named User Plus	Software Update License & Support	Processor License	Software Update License & Support	
Enterprise Edition Options:				
Real Application Clusters	460	101.20	23,000	5,060.00
Real Application Clusters One Node	200	44.00	10,000	2,200.00
Active Data Guard	200	44.00	10,000	2,200.00
Partitioning	230	50.60	11,500	2,530.00
Real Application Testing	230	50.60	11,500	2,530.00
Advanced Compression	230	50.60	11,500	2,530.00
Advanced Security	230	50.60	11,500	2,530.00
Total Options	~200	~50.60	~11,500	~2,530.00

Cláudio Lúcio

15

Motivações

Movimento *NoSQL* - Motivações:

- Evitar mapeamento objeto-relacional
 - Eleva complexidade e custo do desenvolvimento web;
- Movimentos de linguagens de programação e Frameworks para dados
 - Java Persistence API – JPA;
 - Ruby on Rails – RoR;
 - Django – Python;
- Novos paradigmas para processamento paralelo;

Motivações

NoSQL – Mudança de paradigma:

SQL	NoSQL
Dados organizados em tabelas	Dados não são organizados apenas em tabelas: árvores, grafos, pares chave-valor. Melhor estrutura para resolver o problema.
Foco no servidor: I/O, memória, cache e CPU. Abordagem principal: Escalabilidade vertical.	Problema é distribuído. Número de CPU's de acordo com problema. Abordagem principal: Escalabilidade horizontal.
Utiliza mapeamento objeto relacional (ferramentas) para automaticamente gerar SQL(consultas) a partir da camada de objetos	Mapeamento objeto relacional não é necessário. Utiliza documentos/estrutura de dados na aplicação.
Utiliza código procedural e gerenciamento de estado para gerenciamento de transações.	Utiliza programação funcional e algoritmos <i>Map Reduce</i> para particionar o problema em tarefas independentes.

Cláudio Lúcio

17

Motivações

NoSQL – Mudança de paradigma:

SQL	NoSQL
Equipe é treinada em como fazer debug de código procedural e avaliar utilização de servidores.	Equipe é treinada em programação funcional (rapidez) e avalia a transformação dos dados na arquitetura da solução.
Analistas de dados fazem modelos lógicos e físicos para construir esquemas de dados precisos de acordo com os padrões da corporação.	Analistas estão preparados para carregar dados a medida que eles surgem e adaptam esquemas de acordo com a necessidade.
Todas as transações seguem o preceito ACID. Todos os relatórios são consistentes	Utiliza o preceito ACID quando necessário mas o foco é não bloquear escritas. O sistema é eventualmente consistente (dados em processamento).
Bancos de dados são criados em hardware proprietários que são gerenciados por equipe própria.	Bancos de dados são serviços baseado em computação nas nuvens e não requerem equipe de sustentação internas operacionais.



Introdução e contexto NoSQL/Big Data

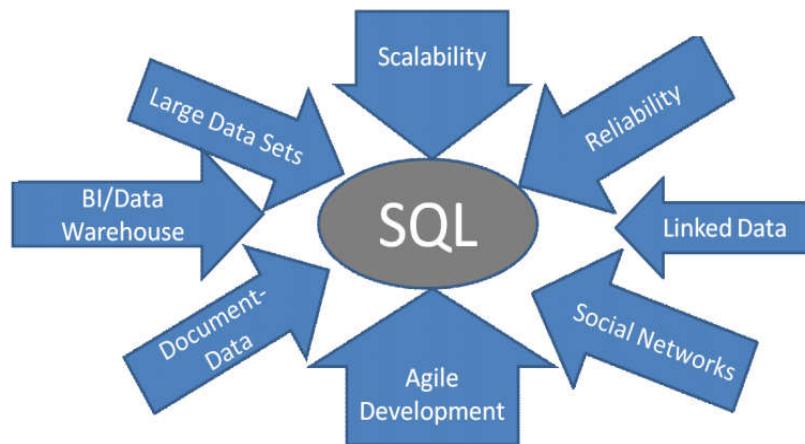
Histórico

Cláudio Lúcio

19

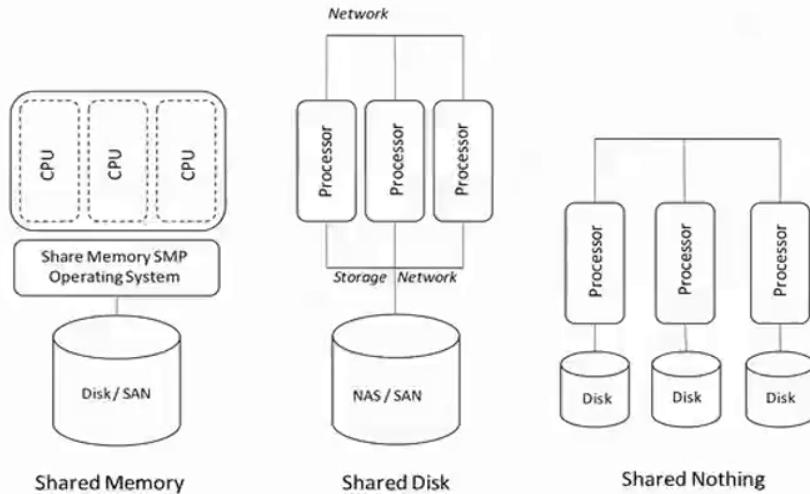
Histórico

Contexto SQL:



Histórico

Contexto SQL:



- SQL executada em paralelo;
- Esquemas de particionamentos (*shared nothing*);
- Tolerância a falhas? E escalabilidade?

Cláudio Lúcio

21

Histórico

Business Intelligence:

- Necessidade de processar históricos;
- Modelagem dimensional;
- OLAP e agregações;
- Cubos;
- Multi-Dimensional eXpressions - MDX , criado pela Microsoft (primeira especificação 1997);
- Alguns não consideram como uma tecnologia NoSQL, pois é centrada na definição de esquemas: tabelas fato, cubos e dimensões

Histórico

Google:

- Artigo do Google sobre o algoritmo *Map Reduce* para indexar páginas da Web utilizando um grande conjunto de máquinas de baixo custo -2004;
- Outro artigo sobre a utilização da plataforma *Big table* para manipular dados estruturados – 2006;

XQuery

- Surge a Xquery como a primeira linguagem padronizada (W3C) para consultar dados estruturados e documentos.
- Utilizada por várias produtos SQL;

23

Histórico

Amazon:

- Apresenta sua implementação para armazenamento distribuído e processamento paralelo: Dynamo;
- Em janeiro de 2012 foi anunciado o DynamoDB (chave-valor) como um serviço web. O serviço promete: escalabilidade, desempenho(utilizam SSD), integridade transacional e baixo custo;

Hadoop

- Projeto da fundação Apache;
- Implementa o *Map Reduce* e DFS(HDFS), adicionando mais uma elevada gama de outros produtos.

Histórico

Resumo:

Evento	Data
Padrão MDX - Microsoft	1997
Artigo Google Map-reduce	2004
Artigo Google BigTable	2006
W3C Xquery	2006
Artigo Amazon Dynamo	2007
Hadoop	2008
Dynamo DB Service - Amazon	2012

Cláudio Lúcio

25

NoSQL
MongoDB
MapReduce
Cassandra
Python
Processamento
BerkeleyDB
Paralelismo
chave-Valor
Documentos
DFS
Colunar

Introdução e contexto
NoSQL/Big Data

Conceitos e Fatos

Conceitos e Fatos

Fatos: Precisamos de outra abordagem

Características de SGBDR:

- Armazenamento orientado para discos locais e estrutura de índices;
 - Mas e a lei de Moore? Considerando memória RAM.
- *Multithreading* para lidar com latência
- Mecanismo de controle de concorrência baseado em *lock*;
- Recuperação baseada em *log (write ahead)*;

Cláudio Lúcio

27

Conceitos e Fatos

Fatos: Precisamos de outra abordagem

As características dos SGBDR não refletem os paradigmas e desafios de computação atuais?

Stonebraker et al. tentou responder esta questão em seu artigo : *The end of an architectural era: (it's time for a complete rewrite)*:

- Desenvolveu um protótipo de BD chamado *H-Store*;
- Foi testado utilizando TPC-C benchmark e mostrou-se 82 vezes mais rápido que solução comercial mais comum;

Conceitos e Fatos

Fatos: Precisamos de outra abordagem

Características da solução de Stonebraker et al., H-Store:

- Executa em um grid computacional;
- Faz uso intensivo de memória (linhas das tabelas, grande parte, em memória);
- Faz particionamento lógico por estação de trabalho;
- Cada estação de trabalho é completamente independente com seus próprios índices (*B-Tree* para armazenamento) e particionamento de memória;
- *Thread* única e transações não são interrompidas – sempre que possível executada em uma única máquina do grid;

Conceitos e Fatos

Fatos: Precisamos de outra abordagem

Características da solução de Stonebraker et al., H-Store:

- Executa apenas transações pré-definidas implementadas como *Stored Procedures*;
- Não faz uso de *redo-log* e sempre tenta evitar escritas do tipo *redo*. Caso seja necessário faz uso de “*transaction commit*”
- Mantém replicas de cada tabela. Comandos de leitura podem acessar qualquer estação do grid e atualizações são feitas em todas as replicas;
- Em resumo: particionamento horizontal, replicação no cluster e campos indexados;

Conceitos e Fatos

Fatos: Precisamos de outra abordagem

Considerações:

- Novos desafios: *Data warehouses*, Bases textuais, bancos de dados científicos(*arrays*),dados semi-estruturados
- Novas abordagens;
- Novos paradigmas;
- Novo “mindset”;



Alexandre Porcelli – DevInVale -2011

Cláudio Lúcio

31

Conceitos e Fatos

Conceitos:

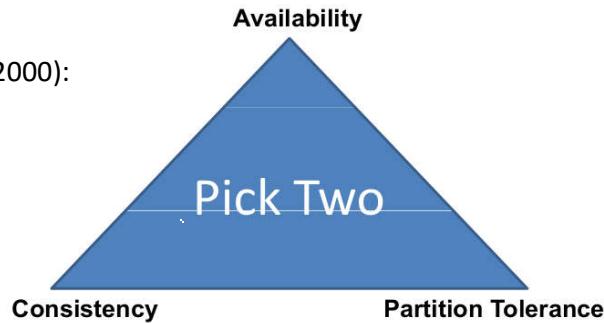
Teorema CAP – Eric Brewer (2000):

- **Consistência**: como sistema fica consistente após uma operação
 - Em um sistema distribuído a consistência acontece se: depois de uma alteração por um 'escritor' a alteração pode ser visualizada por todos os 'leitores';
- **Disponibilidade (*availability*)**: tolerância a falhas
 - Em um sistema distribuído após a falha de qualquer nó, a operação/tarefa global não é afetada;
- **Tolerância a Partição (escalável)**: pode ser entendido como a capacidade de adicionar/remover nós do *cluster* em conjunto com a capacidade de distribuir dados pelos mesmos nós;

Conceitos e Fatos

Conceitos:

CAP Teorema – Eric Brewer (2000):



- Brewer alega que apenas 2 destas características podem ser atendidas em sistemas para compartilhamentos de dados;
- No caso de consistência e particionamento serem requisitos, propriedades **ACID** são necessárias;
- Se disponibilidade e particionamento são favorecidos, então o sistema possui propriedades **BASE**;

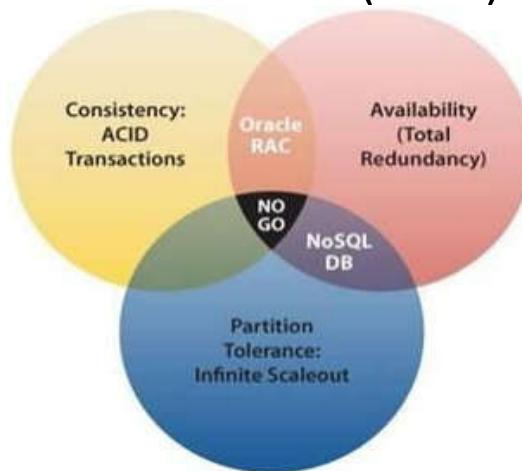
Cláudio Lúcio

33

Conceitos e Fatos

Conceitos:

CAP Teorema – Eric Brewer (2000):



Conceitos e Fatos

Conceitos:

CAP Teorema – Eric Brewer (2000):

- SGBDR: Consistência e disponibilidade
- Amazon's Dynamo: Disponibilidade e particionamento.
Ex.: Atualização de cliente não é imediatamente visualizada pelos outros processos leitores;
- Google's Bigtable: Consistência e disponibilidade;

Cláudio Lúcio

35

Conceitos e Fatos

Conceitos:

Propriedades BASE:

- Brewer:'Basically available', Soft-state', ' Eventual consistency';
- Basicamente disponível: não garante disponibilidade, mas oferece suporte a função A do teorema CAP;
- Estado: indica que os estado do sistema pode mudar a medida que o tempo passa;
- Consistência eventual: sistema não esta consistente a todo momento;

Cláudio Lúcio

Conceitos e Fatos

Conceitos:

ACID x BASE:

ACID	BASE
Strong consistency	Weak consistency – stale data OK
Isolation	Availability first
Focus on "commit"	Best effort
Nested transactions	Approximate answers OK
Availability?	Aggressive (optimistic)
Conservative (pessimistic)	Simpler!
Difficult evolution (e.g. schema)	Faster
	Easier evolution

Table 3.2.: ACID vs. BASE (cf. [Bre00, slide 13])

Cláudio Lúcio

37

Conceitos e Fatos

Conceitos - Consistência:

Propriedades BASE:

Consistência eventual:

- O sistema retorna o último valor escrito;
- Eventualmente pode acontecer de fazer leituras inconsistentes, a medida que atualizações estiverem em progresso (ex.: 500 ms);
- Exemplo: em banco de dados replicado as atualizações são feitas em um 'nó' e serão replicadas para os demais que irão conter as réplicas. Consultas podem ocorrer antes da atualização;

Conceitos e Fatos

Conceitos – Particionamento:

- Para o *Big Data* é necessário utilizar a capacidade de mais de uma única máquina;
- O dado também deve ser replicado para garantir tolerância a falha;
- De acordo com o tamanho da base de dados e necessidade de escalabilidade adota-se as seguintes abordagens para o particionamento:
 - *Memory cache*: utilização de que parte do dado vai para memória RAM. Pode ser utilizada em vários servidores;

Cláudio Lúcio

39

Conceitos e Fatos

Conceitos – Particionamento:

- Cluster: Particionamento entre várias máquinas (críticas para os bancos relacionais em que este conceito foi adaptado – Stonebraker et al.);
- Leitores e escritores separados: Utiliza o esquema de 'mestre/escravo'. O mestre determina máquinas para leitura e outras máquinas para escrita. Os dados são particionados nesta estrutura;
- Sharding: Particionamento dos dados na forma que eles são requisitados e atualizados em conjunto. De forma que fiquem na mesma partição ou nó de execução. Ainda propõe o uso de replicação para tolerância a falhas.



Introdução e contexto NoSQL/Big Data

Abordagens

Cláudio Lúcio

41

Abordagens

Existem várias abordagens de implementação NoSQL:

- Baseado em chave valor;
- Baseado em documentos;
- Baseado em grafos;
- Baseados em colunas;
- Outras;

Abordagens

Baseado em chave valor:

- Estrutura muito simples (binômio): chave e valor;
- Imagine um tabela com dois campos apenas;
- Funções para armazenar as chaves e valores e outra função para resgatar os valores dada uma/várias chave(s) (simplicidade);
- Para estes sistemas há escalabilidade horizontal/tolerância a partição: automaticamente adiciona CPU's e discos aos sistema - “Sharding”;

Cláudio Lúcio

43

Abordagens

Baseado em chave valor:

- É ineficiente quando há o requisito de consistência;
- Problemas para atualização de parte do valor associado a chave;
- Pode ser difícil construir estruturas de dados mais complexas;
- Exemplos: Scalaris, Voldemort, BerkeleyDB e pode ser facilmente implementado em uma linguagem de programação;

Cláudio Lúcio

Abordagens

Baseado em documentos:

- Armazena os dados em uma estrutura de dados de árvore.
Exemplos mais comuns: JSON e XML;
- São extremamente flexíveis (árvores podem ter sub-árvores), e muito utilizadas para ambientes WEB (paradigma natural);
- Documentos podem ser adicionados a qualquer momento na estrutura;
- Suporta grande variabilidade de dados(tipos);
- Acesso aos níveis 'folhas' mesmo em um grande volume de documentos (milhões ou bilhões) pode ter excelente desempenho: utiliza os metadados como documentos e estes são indexados;

Cláudio Lúcio

45

Abordagens

Baseado em documentos:

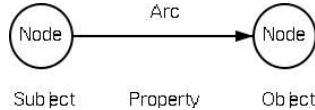
- Armazenam textos diretamente e permitem a utilização de componentes de indexação de textos (ex.:Apache Lucene);
- É capaz de identificar entidades diretamente no texto (descrição) e armazenam estes metadados para rápida recuperação. Ex.: um catálogo de produto pode conter tag's para preço, prazo de garantia, cor e tamanho.
- Alguns autores acreditam que esta tecnologia é a evolução da abordagem chave-valor, pois permite armazenagem de valores agrupados
- Exemplos: MongoDB, CouchDB, RavenDB e outros;

Cláudio Lúcio

Abordagens

Baseados em grafos:

- Estrutura simples, ao invés de duas colunas, utiliza três colunas(nó-arco-nó:unidade básica);
- Também conhecidos como 'Triple-store'
- Capacidade de junção de grafos através dos identificadores dos nós. A mescla de dois grafos é feita automaticamente mesmo se os sistemas são não relacionados: A->B e B->C, logo A->C;



Cláudio Lúcio

47

Abordagens

Baseados em grafos:

- Grafos são a base para alguns sistemas semânticos da WEB que utilizam 'Resource Description Framework' - RDF;
- Estes sistemas usam uma linguagem chamada SPARQL;
- Exemplos: Neo4j, FlockDB e outros;

Abordagens

Baseados em colunas:

- Não são uma novidade. Ex.: Sybase IQ;
- Estruturas colunares armazenam dados de uma coluna;
- As páginas de dados das colunas são armazenadas nas ordens das linhas:

EmpId	Lastname	Firstname	Salary
1	Smith	Joe	40000
2	Jones	Mary	50000
3	Johnson	Cathy	44000

Linha

```
1,Smith,Joe,40000;  
2,Jones,Mary,50000;  
3,Johnson,Cathy,44000;
```

Coluna

```
1,2,3;  
Smith,Jones,Johnson;  
Joe,Mary,Cathy;  
40000,50000,44000;
```

Abordagens

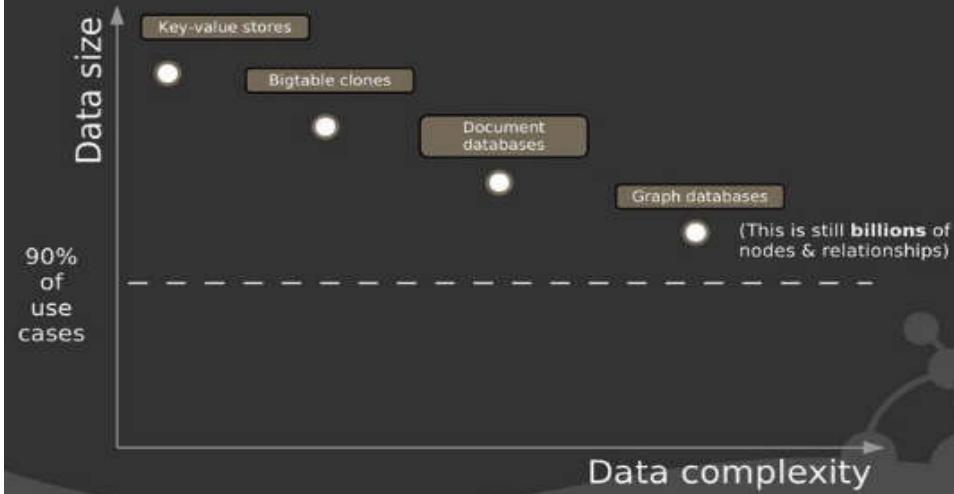
Baseados em Colunas:

- As consultas são processadas de forma diferente:
 - Exemplo: Uma consulta com 3 colunas implica que 3 estruturas de colunas são acessadas, os resultados são mesclados pelo ID da linha. Resultado é materializado;
- A ideia básica dos bancos de dados colunares é minimizar o tráfego de dados (Disco e CPU):
 - As consultas não precisam de linhas completas (todas as colunas) o que sobrecarrega o sistema;
 - Usando apenas as colunas necessárias há a minimização de utilização de recursos;
- Muitos bancos de dados por linhas estão buscando estas opções: SQL SERVER 2012;
- Exemplos: HBase, Cassandra e outros

Abordagens

Recomendações:

NOSQL data models

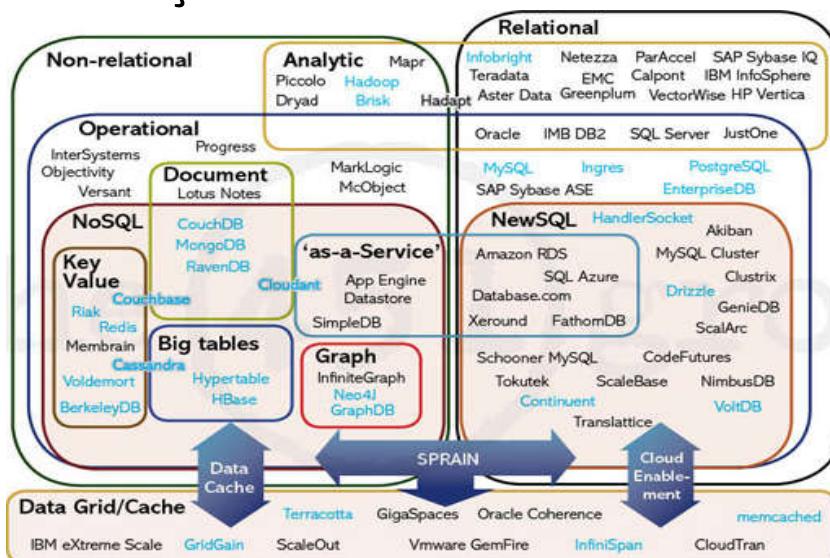


Cláudio Lúcio

51

Abordagens

Recomendações:



Luciano Ramalho – DevInVale -2011

Abordagens

Recomendações:

- Para introdução desta tecnologia recomenda-se começar com um projeto piloto;
- Este tipo de projeto tem a meta de desmitificar a tecnologia na organização;
- Algumas dicas:
 - Encontre um patrocinador(área de negócio) que entenda as necessidades e características da base de dados e tenha interesses: custo, Big Data, agilidade;
 - Avalie o tamanho do projeto e verifique se o uso de NoSQL realmente será um benefício;

Cláudio Lúcio

53

Abordagens

Recomendações:

- Alinhas as expectativas com todos os envolvidos no projeto;
- Avalie se não é necessário envolver consultoria externa com conhecimento na abordagem e produtos;
- Procure envolver mais pessoas da equipe de TI durante todo o processo de desenvolvimento da solução. Lembre-se que não haverá necessidade de mapeamento objeto-relacional e isto pode simplificar o processo.
- Pense em uma estratégia para comunicar todo o projeto;

Cláudio Lúcio

Atividade

Atividade:

Leitura do texto Big Data;

Definição de grupos para o seminário

- Reflexão
 - O **Big Data** traz a necessidade de novas abordagens para processamento dos dados. Entendimento dos 4 v's do BigData.
 - O que é o teorema **CAP** e quais as premissas **BASE** para sistemas distribuídos?
 - Qual o real significado de **NoSQL**? Eles faz sentido para você ou sua realidade?
 - Devemos evitar a abordagem “**One size fits all**” ou bala de prata em bancos de dados?
 - Quais os principais tipos de **abordagens tecnológicas** o movimento NoSQL propõem?



Fragments extraídas do texto:

Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data

Marcos Rodrigues Vieira¹, Josiel Maimone de Figueiredo², Gustavo Liberatti²,
Alvaro Fellipe Mendes Viebrantz²

¹IBM Research Laboratory - Brazil
mvieira@br.ibm.com

²Instituto de Computação
Universidade Federal de Mato Grosso (UFMT)
josiel@ic.ufmt.br, {liberatti.gustavo, alvarowlfx}@gmail.com

Um dos grandes desafios atualmente na área de Computação é a manipulação e processamento de grande quantidade de dados no contexto de Big Data. O conceito Big Data pode ser resumidamente definido como uma coleção de bases de dados tão complexa e volumosa que se torna muito difícil (ou impossível) e complexa fazer algumas operações simples (e.g., remoção, ordenação, summarização) de forma eficiente utilizando Sistemas Gerenciadores de Bases de Dados (SGBD) tradicionais. Por causa desse problema, e outros demais, um novo conjunto de plataformas de ferramentas voltadas para Big Data tem sido propostas, como por exemplo Apache Hadoop [3].

A quantidade de dados gerada diariamente em vários domínios de aplicação como, por exemplo, da Web, rede sociais, redes de sensores, dados de sensoriamento, entre diversos outros, estão na ordem de algumas dezenas, ou centenas, de Terabytes. Essa imensa quantidade de dados gerados traz novos grandes desafios na forma de manipulação, armazenamento e processamento de consultas em várias áreas de computação, e em especial na área de bases de dados, mineração de dados e recuperação de informação. Nesse contexto, os SGBD tradicionais não são os mais adequados, ou “completos”, às necessidades do domínio do problema de Big Data, como por exemplo: execução de consultas com baixa latência, tratamento de grandes volumes de dados, escalabilidade elástica horizontal, suporte a modelos flexíveis de armazenamento de dados, e suporte simples a replicação e distribuição dos dados.

.....

1.2.1. Big Data

O termo Big Data é bem amplo e ainda não existe um consenso comum em sua definição. Porém, Big Data pode ser resumidamente definido como o processamento (eficiente e escalável) analítico de grande volumes de dados complexos produzidos por (várias) aplicações. Exemplos de aplicações no contexto Big Data varia bastante, como aplicações científicas e de engenharias, redes sociais, redes de sensores, dados de Web Click, dados médicos e biológicos, transações de comércio eletrônico e financeiros, entre inúmeras outras. As semelhanças entre os dados desses exemplos de aplicações incluem: grande quantidade de dados distribuídos, características de escalabilidade sob demanda, operações ETL (Extract, Transform, Load) de dados “brutos” (raw) semi- ou não estruturados para dados estruturados e, a necessidade de extrair conhecimento da grande quantidade de dados.

Três fatores influenciaram o grande aumento de volume de dados sendo coletados e

armazenados para posterior análise: difusão dos dispositivos captação de dados, dispositivo com armazenamento na ordem de Terabytes e aumento de velocidade de transmissão nas redes. Os dispositivos de aquisição, bem como os dispositivos de armazenamento de grande escala se difundiram principalmente pelo seu barateamento (e.g., redes de sensores, GPS, smartphones), enquanto que as redes aumentaram sua velocidade e abrangência geográfica. Outro fator importante é a facilidade de geração e aquisição de dados gerados digitalmente, como máquinas fotográficas digitais, smartphones, GPS, etc. Como consequência novas demandas estão surgindo, como a demanda por análise de grande volume de dados em tempo real (data analytics), o aumento do detalhamento das informações, bem como plataformas escaláveis e eficientes de baixo custo.

Basicamente, podemos resumir as características do contexto Big Data em quatro propriedades: (1) dados na ordem de dezenas ou centenas de Terabytes (podendo chegar a ordem de Petabytes), (2) poder de crescimento elástico, (3) distribuição do processamento dos dados; e (4) tipos de dados variados, complexos e/ou semiestruturados. A característica de análise dos dados na ordem de Terabytes envolve, entre outros aspectos, o requisito de alto poder computacional de armazenamento e processamento dos dados. A elasticidade está relacionada ao fato de que a quantidade de dados pode variar de alguns Megabytes a vários Terabytes (e vice-versa) em um espaço de tempo relativamente curto, fazendo com que a estrutura de software/hardware adapte-se sob demanda, i.e. seja alocada/desalocada dinamicamente. A distribuição significa que os dados devem ser distribuídos de forma transparente em vários nós espalhados de processamento, o que demanda armazenamento, processamento e controle de falhas distribuído. Finalmente, a quarta característica está relacionada a adoção de modelos mais apropriados, flexíveis e eficientes para o armazenamento de tipos de dados variados e semi-estruturados. Vale ressaltar que o modelo relacional não é o mais adequado pois não possui flexibilidade para o armazenamento de dados e evolução no modelo para tais tipos de dados citados acima.

A análise de dados (data analytics) no contexto de Big Data normalmente envolve processamento da ordem de Terabytes em dados de baixo valor (i.e., informação original “bruta”) que são transformados para dados de maior valor (e.g., valores agregados/sumarizados). Mesmo com a grande quantidade de dados Big Data em si não garante a qualidade da informação, pois a análise continua, em grande parte, sendo muito subjetiva. Isso se deve ao fato que os dados em si não são autoexplicativos, onde o processo de limpeza, amostragem, e relacionamento dos dados continua sendo crítico e passível a erros, aproximações e incertezas [14]. Por exemplo, a análise de dados da ordem de Petabytes (ou Exabytes) de cunho científicos (e.g., dados genômicos, física ambiental e simulações numéricas) tem se tornado muito comum hoje em dia, onde é aceitável que o resultado da análise contenham imprecisão (i.e., erro entre certos limites de erros), porém seja computado de forma (relativamente) rápida e/ou em tempo real.

Recentemente, ambientes de computação em nuvem (cloud computing) têm sido utilizados para o gerenciamento de dados em forma de Big Data, enfocando principalmente em duas tecnologias: Bases de Dados Como Serviço (Database as a Service (DaaS)) e Infraestrutura Como Serviço (Infrastructure as a service (IaaS)) (para maiores detalhes). DaaS utiliza um conjunto de ferramentas que permite o gerenciamento remoto dos servidores de dados mantidos por uma infraestrutura externa sob demanda. Essa infraestrutura IaaS fornece elasticidade, pagamento sob demanda, backup automáticos, e rapidez de implantação e entrega.

As principais características que envolvem os ambientes em nuvem são: escalabilidade, elasticidade, tolerância a falhas, auto gerenciamento e a possibilidade de funcionar em hardware commodity (comum). Por outro lado, a maioria dos primeiros SGBD relacionais comerciais foram desenvolvidos e concebidos para execução em ambientes corporativos. Em um ambiente de computação em nuvem traz inúmeros desafios do ponto de vista computacional. Por exemplo, o controle de transação na forma tradicional (i.e., definida pelas propriedades ACID) em um ambiente de nuvem é extremamente complexo.

De uma maneira geral, os ambientes em nuvem precisam ter a capacidade de suportar alta carga de atualizações e processos de análises de dados. Os data centers são uma das bases da computação em nuvem, pois uma grande estrutura como serviço escalável e dinâmica é fornecida para vários clientes. Um ambiente de gerenciamento de dados escalável (scalable data management) pode ser dividido em: (1) uma aplicação complexa com um SGBD gerenciando uma grande quantidade de dados (single tenant); e (2) uma aplicação no qual o ambiente deve suportar um grande número de aplicações com dados possivelmente não muito volumosos [2]. A influência direta dessas duas características é que o SGBD deve fornecer um mesmo esquema genérico para inúmeros clientes com diferentes aplicações, termo denominado bases de dados multitenant.

É importante lembrar que em ambientes multitenant a soma dos tenant pode gerar um grande volume de dados armazenado no SGBD. Este característica é apropriadamente gerenciada pelo ambiente em nuvem, onde novas instâncias de SGBD são criadas em novos nós e/ou servidores do ambiente (os dados de diferentes tenant são independentes entre si).

Em ambientes tradicionais, quando uma aplicação cresce sua complexidade o SGBD atende às requisições e a escalabilidade do sistema no modo que aumenta o seu poder computacional. Por exemplo, o poder computacional do sistema como um todo cresce a medida que mais memória e/ou número de nós do cluster são adicionados ao servidor. No entanto, esta abordagem são caras, dependentes de arquitetura, e normalmente não presentes em SGBD livres (open sources).

A Revolução do Big Data – Jornal O Globo 02/07/2012

Você vai ao mercado com o objetivo de comprar apenas o que falta para o jantar e, ao passar pelo corredor de produtos de higiene, seu celular o surpreende com uma mensagem. O remetente é a própria varejista, que deseja chamar sua atenção para o desodorante em promoção na prateleira ali do lado. O SMS não diz, mas a rede sabe que o seu estoque do produto está mesmo no fim e que, há duas semanas, você escreveu no Facebook o quanto gostava da marca. Se a precisão da mensagem lhe é espantosa, prepare-se: a tecnologia que cruza dados dessa forma já existe, representa um mercado direto estimado em US\$ 70 bilhões e está invadindo empresas e governos no Brasil e no mundo — o que deve elevar à enésima potência a possibilidade de ganhos com o uso dessas informações. A promessa é de revolução em várias áreas da economia e até na ciência — além de uma renovada discussão sobre privacidade.

Trata-se do Big Data, termo de mercado para o conjunto de soluções que analisa informações em variedade, volume e velocidade inéditos até hoje. Ferramentas desse tipo surgiram no fim da década passada, mas este ano o conceito extrapolou de vez os limites da academia e dos setores de TI. Isso porque o preço para armazenamento de dados está despencando e diversas ferramentas baratas ou gratuitas para lidar com tamanho volume informações estão surgindo.

O uso dessa nova tecnologia tem vasta abrangência. No último Fórum Econômico Mundial, em Davos, foi publicado um estudo mostrando como o Big Data pode ser uma arma contra problemas socioeconômicos. E até Brad Pitt tem contribuído para sua popularização: o filme "Moneyball (O Homem que mudou o jogo)", que protagoniza, conta a história da mais famosa aplicação do conceito: o gerente de um time de beisebol que usa o Big Data para reunir um elenco de primeira linha sem gastar muito.

"Pré-sal existe por causa do Big Data"

O executivo de operações da EMC, Pat Gelsinger, diz que o mercado global de Big Data já movimenta US\$ 70 bilhões por ano — sem contar inestimáveis ganhos nos negócios. A consultoria IDC estima que o segmento crescerá quase 40% ao ano entre 2010 e 2015, mas considera um patamar de US\$16,9 bilhões ao fim desse período. A tecnologia envolve tanto dinheiro porque soluciona um problema inadiável para a economia, o da quantidade de dados digitais. O volume deve crescer do atual 1,8 zettabyte para 7,9 zettabytes em 2015, prevê a IDC. Zettabyte equivale a um trilhão de gigabytes.

A centelha dessa revolução é a proliferação de plataformas que geram dados como nunca. São celulares, GPS, redes sociais, câmeras e sensores dos mais diversos tipos. Grande parte da informação gerada é classificada de não-estruturada: ou seja, não é facilmente computável, costuma ser criada pelo ser humano, não por máquina. Até pouco tempo, essa informação só podia ser compreendida por pessoas. Com o Big Data, as máquinas aprendem a lê-la. Essa é, nas palavras de especialistas, a beleza do conceito.

Nos últimos 50 anos, a evolução do mercado de TI se deu apenas no "T" da sigla, a tecnologia. Com o Big Data, é chegada a hora de o "I", de inteligência, guiar o avanço — afirmou Alexandre Kazuki, diretor de marketing da divisão da HP Brasil que cuida de Big Data.

Se o Big Data está dando os primeiros passos no mundo, a tecnologia ainda engatinha no Brasil, na avaliação de Kátia Vaskys, diretora de Business Analytics da IBM. Ela cita a forma como a maioria das empresas brasileiras monitora suas marcas nas redes sociais:

- Aqui costuma-se contratar um time de estagiários para isso. Isso é basear a estratégia de marketing na intuição, mas não há intuição que resista a tanta informação! Há uma ferramenta tecnológica para fazer isso com muito mais precisão e em tempo real.
- A aplicação por aqui está restrita a setores como varejo, financeiro (análise de risco),

telecomunicações e petrolífero, mas começa a chegar à mídia.

- A Renner usa o Big Data para monitorar em tempo real o fluxo de mercadorias da loja ao cruzar dados de localização GPS dos caminhões dos fornecedores com os níveis dos estoques, contou o diretor de TI Leandro Balbinot. A rede também acompanha a aceitação dos produtos nas redes sociais. E já imagina outros usos, como a possibilidade de reorganizar uma loja com base em dados meteorológicos. Exemplo: se, nas últimas chuvas, os clientes compraram mais calças ou acessórios, a rede pode dar destaque a esses itens quando os primeiros pingos caírem.

Apesar de o uso no Brasil ainda ser pouco maduro, a expectativa é enorme. Temos um dos principais mercados de internet no mundo, sobretudo de redes sociais, o que é crucial para a adesão ao Big Data — observou Maurício Prado, gerente geral de servidores da Microsoft Brasil.

Só sabemos que o pré-sal existe por causa do Big Data e da economia da nuvem — resumiu Patrícia Florissi, CTO da EMC para as Américas.

Isso porque a tecnologia agiliza o processamento de dados sísmicos captados pelas sondas que procuram petróleo no fundo do mar. Como são milhões as variáveis, o trabalho exige intermináveis simulações de imagens, e só o Big Data é capaz de dar conta do trabalho em um tempo razoável.

Visando a esse mercado, a gigante EMC está construindo no Parque Tecnológico do Fundão um centro de pesquisas totalmente dedicado ao uso de Big Data para a indústria do petróleo. Ele ficará pronto em no máximo dois anos e empregará 35 pesquisadores. A companhia vai investir R\$ 100 milhões no país nos próximos quatro anos.

Polícia chega antes do crime

Há também iniciativas brasileiras na seara governamental, aceleradas pela proximidade da Lei de Acesso à Informação, que entra em vigor em maio. Uma parceria do Ministério do Planejamento, do Serviço Federal de Processamento de Dados (Serpro) e da PUC Rio disponibilizou na web dados abertos dos mandatos do governo Lula.

A massificação do Big Data, porém, enfrenta obstáculos. O maior deles é com a privacidade. Mas, para Karin Breitman, da PUC-Rio, os cientistas não devem "censurar" pesquisas:

É uma questão ética. Cabe à sociedade impor limites à aplicação da ciência e da tecnologia, mas os pesquisadores precisam trabalhar na ponta.

Outro problema é a escassez de profissionais com habilidades em matemática, estatística e computação. O Big Data levou as empresas a uma disputa frenética por esse perfil e tornou a IBM a maior empregadora de matemáticos PhDs no mundo. O instituto McKinsey prevê que faltarão até 190 mil desses profissionais em 2018 nos EUA.

Já há carência desse profissional no Brasil. Se houver uma explosão do Big Data, teremos problemas — advertiu Kazuki, da HP.

Apesar dos desafios, a expectativa é enorme. A "Economist" escreveu que o Big Data pode transformar modelos de negócio de empresas centenárias. A RollsRoyce, cita, deixaria de vender turbinas para alugá-las, cobrando pelo uso. Sensores e o histórico do cliente dariam o preço.

Patrícia Florissi, da EMC, diz que ainda é incipiente o uso da presciência da tecnologia. Por exemplo: como são capazes de entender imagens, softwares de Big Data poderiam monitorar as câmeras de uma cidade e acionar a polícia antes de um crime acontecer com base em padrões que antecedem assaltos e assassinatos. Sairíamos de "Moneyball" para cair em "Minority Report" — com os prós e contras disso.

A NOVIDADE

A grande novidade das soluções de Big Data é lidar também com os chamados dados não-estruturados, que até então só podiam ser compreendidos por pessoas. São tweets, posts no Facebook, vídeos, geolocalização e comportamentos de clientes que dependem de contexto para ter sentido.

Esses dados não-estruturados representam 85% das informações com as quais as empresas lidam hoje

A quantidade global de dados digitais deve crescer de 1,8 zettabyte, hoje, para 7,9 zettabytes em 2015. Daqui a três anos, toda a informação do mundo poderá ser armazenada em:

85%

O mercado de Big Data crescerá quase 40% ao ano até 2015

493 bilhões de iPads

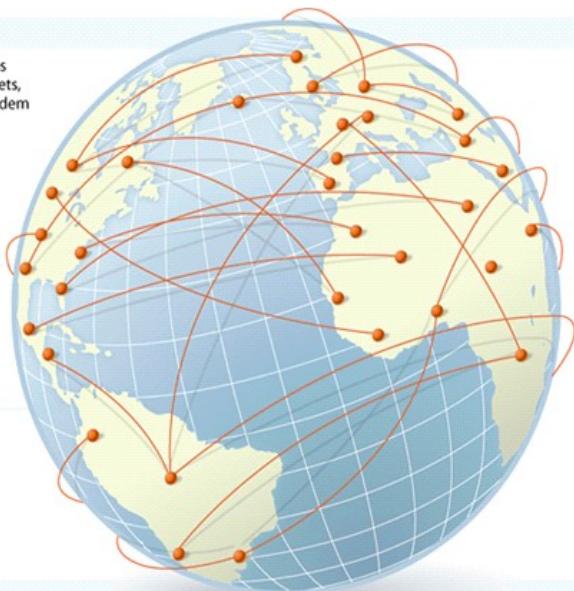
COMPARE

1 Zettabyte é igual

1.000.000.000.000.000.000.000 bytes

1 Gigabyte é igual

1.000.000.000 bytes



Alguns exemplos de como a solução tem sido usada

A companhia Skybox tira fotos de satélite e vende a seus clientes informações em tempo real sobre a disponibilidades de vagas de estacionamento livres numa cidade em determinada hora ou quantos navios estão ancorados no mundo neste momento

O projeto Global Pulse, das Nações Unidas, vai utilizar um programa que decifra a linguagem humana na análise de mensagens de texto e posts em redes sociais para prever o aumento do desemprego, o esfriamento econômico e epidemias de doenças

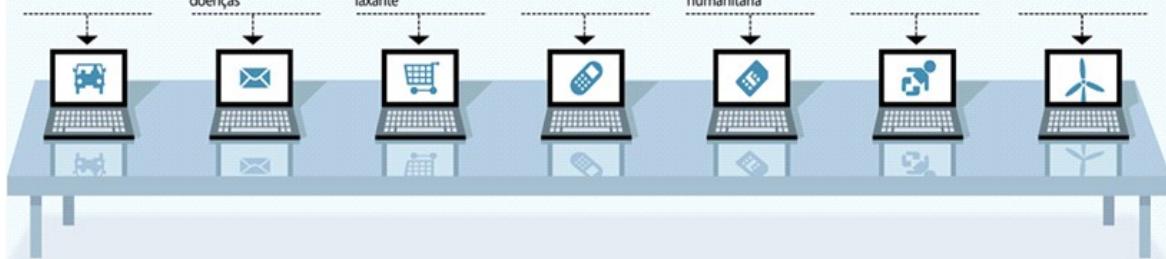
A varejista americana Dollar General monitora as combinações de produtos que seus clientes põem nos carrinhos. Ganhou eficácia e ainda descobriu curiosidades: quem bebe Gatorade tem mais chances de comprar também laxante

A Sprint Nextel saltou da última para a primeira posição no ranking de satisfação dos usuários de celular nos EUA ao integrar os dados de todos os seus canais de relacionamento. De quebra, cortou pela metade os gastos com call center

No terremoto do Haiti, pesquisadores americanos perceberam antes de todo mundo a diáspora de Porto Príncipe por meio dos dados de geolocalização de 2 milhões de chips SIM de celulares, facilitando a atuação da ajuda humanitária

Um hospital no Canadá usou tecnologia da IBM e da Universidade de Ontário para monitorar em tempo real dezenas de indicadores de saúde de bebês prematuros. O cruzamento permitiu aos médicos antecipar ameaças às vidas das crianças

Em busca dos melhores lugares para instalar turbinas eólicas, a dinamarquesa Vestas Wind analisou petabytes de dados climáticos, de nível das marés, mapas de desmatamento etc. O que costumava levar semanas durou algumas horas





Nome: _____ Data: _____

1. Leia o texto: “A revolução do Big Data”.
2. De acordo com o texto, responda:
 1. Com suas palavras descreva o termo Big Data aplicado ao contexto Brasileiro?
 2. A tecnologia *BigData* faz sentido nos negócios da sua empresa? Cite um exemplo de aplicação? Se não faz sentido, cite uma aplicação do conceito em alguma área de negócio que você conhece.
 3. Cite possíveis vantagens para os negócios na utilização de Big Data:
 4. Você acredita que os profissionais de TI estão preparados para lidar com o Big Data. Justifique.

NoSQL MongoDB MapReduce Cassandra Python Processamento BerkeleyDB Paralelismo DFS chave-Valor Documentos Colunar

Paradigmas de computação distribuída e bancos de dados

Cláudio Lúcio

1

Paradigmas de computação e bancos de dados

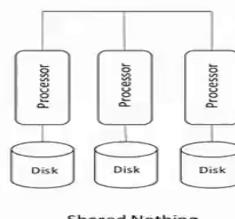
Agenda

- Sistemas de arquivos distribuídos
- Algoritmo *Map Reduce*
- Consistência em bancos de dados não relacionais

Paradigmas de computação e bancos de dados

Muitas soluções NoSQL buscam:

- Resolver problemas *Big Data*;
- Demandar estrutura de computação paralela;
- Esquemas de particionamentos (*shared nothing*);



- Tolerância a falhas;
- Escalabilidade

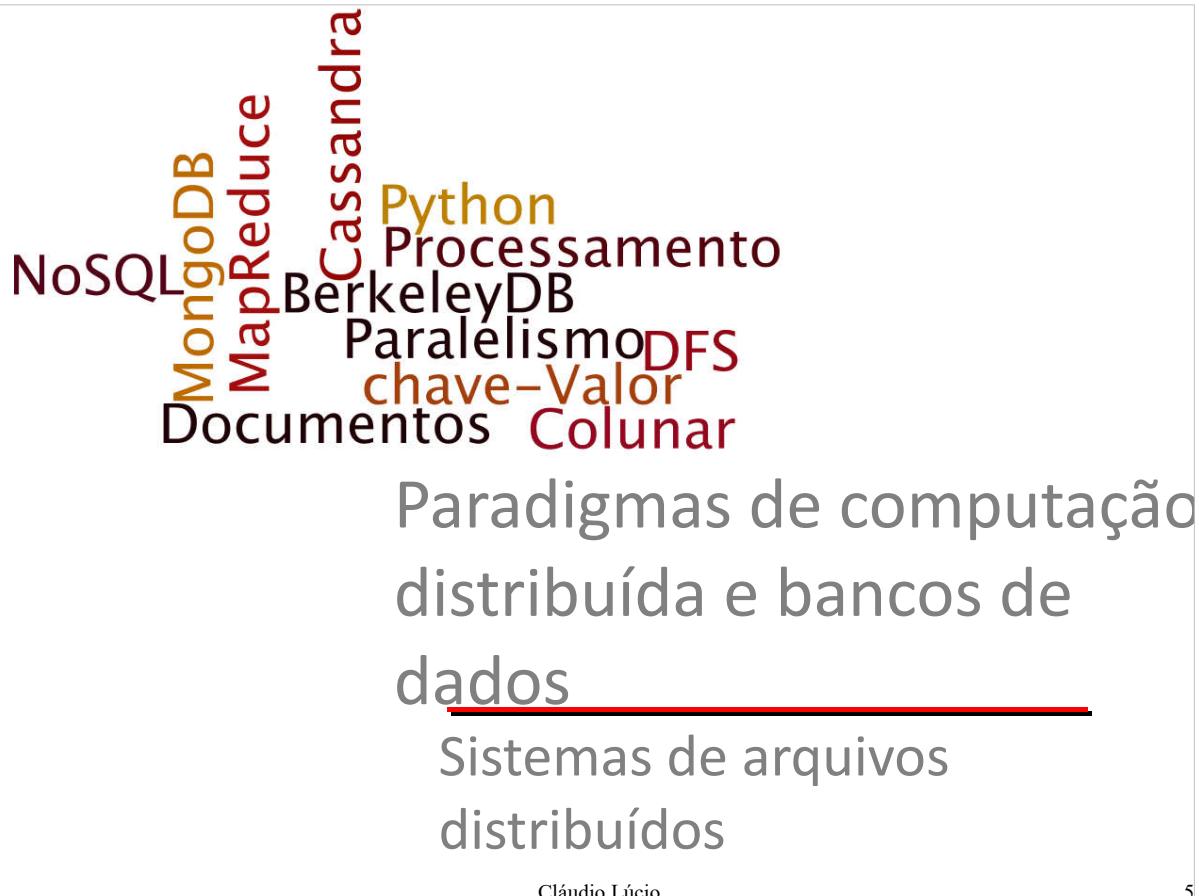
Cláudio Lúcio

3

Paradigmas de computação e bancos de dados

Alguns paradigmas de computação são fundamentais:

- Várias soluções vão utilizar sistemas de arquivos distribuídos;
- Esquemas de controle de rélicas e consistência dos dados;
- Algoritmos para execução paralela em conjunto com estruturas de arquivos;



Sistemas de arquivos distribuídos

Histórico:

- Processamento intensivo era feito em hardware especializado (processadores, cache, discos e memória);
- A Web e o *Big Data* exigem processamento intensivo, mas em outra estrutura de hardware:
 - Centenas ou milhares de computadores em rede (nós);
 - Operação destes computadores de forma mais ou menos independente;
 - Cada um dos nós é um '*commodity hardware*' – custo reduzido;
 - A estrutura em geral é tolerante a falhas;
 - Utilizam sistemas de arquivos especializados;

Sistemas de arquivos distribuídos

Organização física da estrutura:

- A organização física destas máquinas pode seguir este exemplo:
 - Nós são armazenados em *racks* (8-64 em um *rack*);
 - Os nós em um *rack* são conectados via rede (*gigabit Ethernet*);
 - Conjuntos de *racks* são disponíveis na estrutura formando uma espécie de *cluster*;
 - A conexão entre os *racks* também pode ser otimizada;
 - Quanto maior o número de *racks* ou nós, maior a probabilidade falha (de um dos nós);

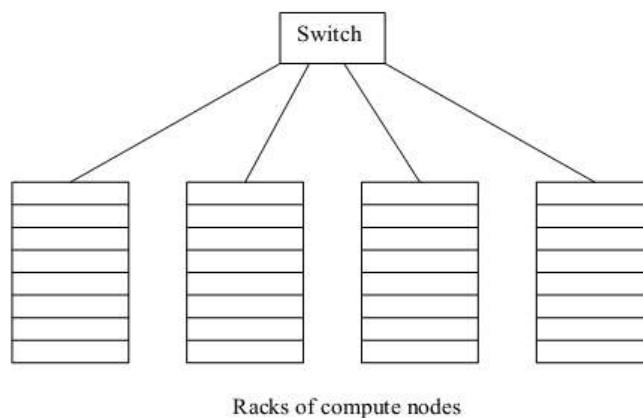
Cláudio Lúcio

7

Sistemas de arquivos distribuídos

Organização física da estrutura:

- A organização física destas máquinas pode seguir este exemplo:



Sistemas de arquivos distribuídos

Computação nesta estrutura:

- Cálculos computacionais nesta estrutura podem levar minutos ou mesmo horas;
- Os cálculos não podem ser reiniciados toda vez que um componente (*rack* ou nó de execução) falha;
- Proposta de solução:
 - Arquivos armazenados de forma redundante (*Distributed File System - DFS*);
 - Cálculos devem ser divididos entre os nós, de forma que se algum nó falhar, somente o trabalho atribuído ao nó deve ser reexecutado;

Cláudio Lúcio

9

Sistemas de arquivos distribuídos

Características DFS:

- Arquivos devem ser 'grandes', gigabytes, pelo menos; Arquivos menores não fazem sentido no *DFS*;
- Arquivos no *DFS* são raramente atualizados (*write-once-read-many*). Adicionalmente dados são adicionados para os arquivos (periodicidade, processamento batch);
- Arquivos são divididos em partes ('*chunks*' ou blocos), normalmente 64 megabytes e replicados para, pelo menos, 3 nós (em *racks* diferentes);
- Os Itens acima são customizáveis;

Sistemas de arquivos distribuídos

Características DFS:

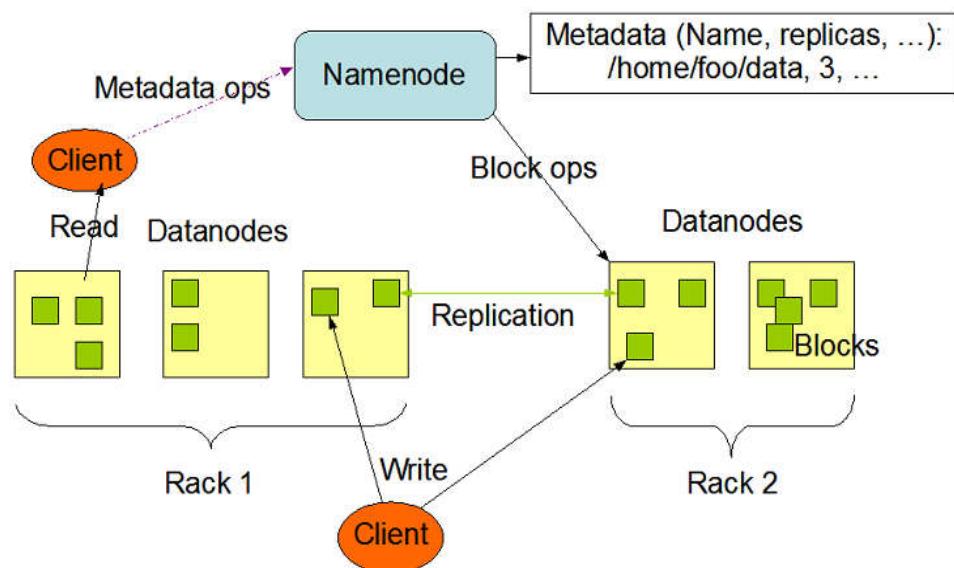
- As informações dos blocos e replicas é controlado utilizando metadados e com um figura central no cluster: '*name node*' ou '*master node*';
- *Name node*:
 - Gerencia o sistema de arquivos(rélicas, blocos, nós e *racks*): abrir, fechar, renomear arquivos;
 - Gerencia o acesso dos clientes ao arquivos;
- Os outros nós do *cluster* são chamados de '*data node*' ou '*slave node*';
 - Executam as operações enviados pelo '*Name node*': criação, exclusão e replicação de blocos;

Cláudio Lúcio

11

Sistemas de arquivos distribuídos

Características DFS:



Sistemas de arquivos distribuídos

Características DFS:

- Possuem regras de sistemas de arquivos: *rack*, '*data node*', *namespaces*, diretórios e arquivos;
- Além disto o DFS gerencia os blocos e sua distribuição/replicação nos '*data nodes*';
- Padrão de réplicas 1/3(forá do rack) e 2/3(no rack);
- O '*name node*' periodicamente recebe um relatório de blocos do '*data node*';

Cláudio Lúcio

13

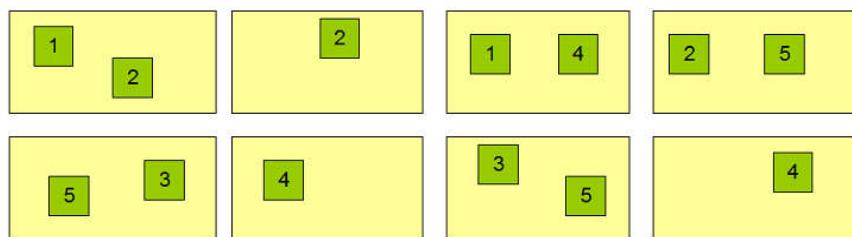
Sistemas de arquivos distribuídos

Características DFS:

Block Replication

```
Namenode (Filename, numReplicas, block-ids, ...)  
/users/sameerp/data/part-0, r:2, {1,3}, ...  
/users/sameerp/data/part-1, r:3, {2,4,5}, ...
```

Datanodes



Sistemas de arquivos distribuídos

Características DFS:

- O '*name node*' é replicado para outra estrutura no *DFS* e também possui *log* de alterações;
- A estrutura de arquivos é, tipicamente, mantida em memória;
- '*Name node*' também faz rebalanceamento de carga: elevada demanda para um bloco, por exemplo;

Cláudio Lúcio

15

Sistemas de arquivos distribuídos

Implementações DFS:

- '*Google File System*' (*GFS*), implementação original e primeira a ficar 'famosa';
- '*Hadoop Distributed File System (HDFS)*', *open-source*. É disponibilizada pela fundação apache e implementada em java;
- CloudStore, é outra implementação *DFS open-source*, originalmente desenvolvida pela Kosmix.

Paradigmas de computação distribuída e bancos de dados

Algoritmo Map Reduce

Cláudio Lúcio

17

Algoritmo Map Reduce

Origens:

- Patente original é do Google, mas é utilizado em várias outros sistemas de computação paralela;
- A ideia é derivada da programação funcional:
 - *Map* e *reduce* são dois tipos de funções comuns;
 - *Map*:
 - Aplica um função ou operação para cada elemento em uma lista;
Ex.: multiplicação por 2;
[1,2,3,4] Map function → [2,4,6,8]
 - Não altera o dado original. Evita o princípio '*Shared Data*';
 - Pode ser executado de forma paralela;

Algoritmo Map Reduce

Origens:

- A ideia é derivada da programação funcional:
 - *Reduce*:
 - É uma função de agrupamento ou compressão;
 - Aplica uma função em conjunto de dados reduzindo para um simples valor;
 - Pode ser executado de forma paralela;
 - Ex.: [2,4,6,8,] → Reduce function → [20]
 - De forma geral:
 - O algoritmo pode ser usado sempre que houver uma lista;
 - Para cada elemento da lista uma função que a transforme;
 - Outra função que possa ser aplicada ao conjunto de dados transformados de forma a agregá-los;

Cláudio Lúcio

19

Algoritmo Map Reduce

Detalhes de funcionamento:

- A implementação do algoritmo é utilizada para realizar computação no *DFS* para arquivos 'grandes' e com execução tolerante a falha;
- É necessário escrever as duas funções: *Map* e *reduce*;
- O sistema lida com os demais detalhes:
 - Execução paralela;
 - Coordenação de tarefas (*Map* e *reduce*);
 - Lidar com a tolerância a falhas;

Algoritmo Map Reduce

Detalhes de funcionamento:

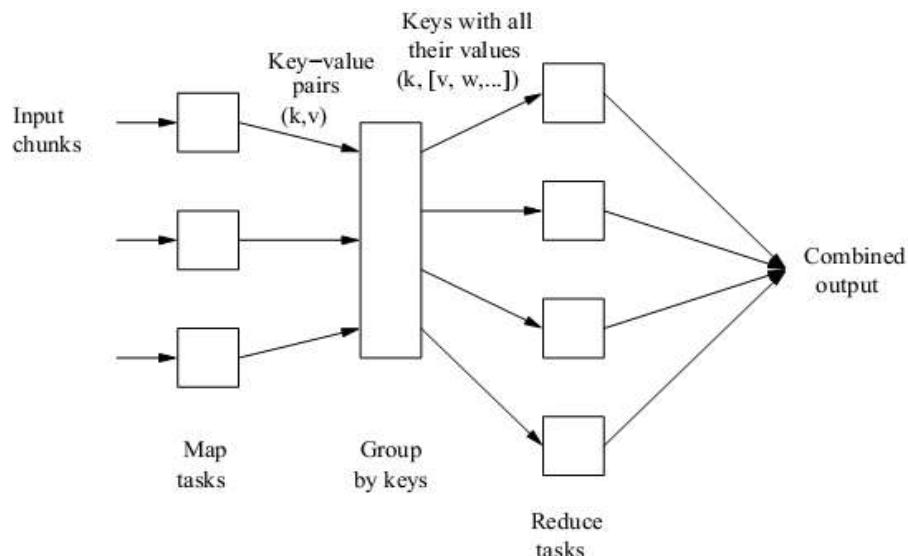
- Seguintes passos de execução:
 - Um arquivo é armazenado no *DFS* com vários blocos em vários nós e *racks*;
 - Um conjunto de tarefas do tipo *Map* é criado, para cada *Map* existe um ou mais blocos que serão processados; As tarefas *Map* vão transformar o dado em um estrutura chave valor ou tuplas;
 - As estruturas chave valor são coletadas pelo controlador *master* e ordenadas pelas suas chaves;
 - As chaves serão agrupadas e divididas para as tarefas do tipo *Reduce* (uma chave, com vários valores será processado por uma e só uma tarefa *Reduce*);
 - As tarefas do tipo *Reduce* vão então agrupar os dados pelas chave, uma por vez.

Cláudio Lúcio

21

Algoritmo Map Reduce

Detalhes de funcionamento:



Algoritmo Map Reduce

Detalhes de funcionamento:

Tarefas *Map*:

- Um bloco no DFS possui vários 'membros' que serão processados. Cada membro só pertence a apenas 1 bloco;
- A estrutura chave valor é importante pois permite a execução de várias tarefas *Map* em paralelo;
- A função *Map* é responsável por converter os dados para a estrutura chave valor;
- Chaves não são 'chave', no sentido estrito, não precisam ser únicas;

Cláudio Lúcio

23

Algoritmo Map Reduce

Detalhes de funcionamento:

Tarefas *Map*:

- Exemplo clássico - Contar o número de palavras em uma coleção de documentos:
 - Cada tarefa *Map* lê um documento ou vários documentos;
 - Cada palavra será considerada uma chave: w_1, w_2, \dots, w_n ;
 - A seguinte estrutura chave valor será criada:

$\langle w_1, 1 \rangle$

$\langle w_2, 1 \rangle$

\dots

$\langle w_n, 1 \rangle$

Sugestões para melhorar a função *Map* ???

Algoritmo Map Reduce

Detalhes de funcionamento:

Agrupamento e agregação:

- São realizados, sempre da mesma forma, independente do que as funções *Map* e *Reduce* façam;
- O “*Name node*” controla este processo:
 - O número de tarefas *reduce* já são conhecidos: r (pode ser previamente determinada);
 - Cria um função *hash* (0 até $r-1$) que é aplicada nas chaves;
 - Cada chave gerada pela tarefa *Map* é então gravada em um dos r arquivos locais;
 - Após todas as tarefas *map* serem finalizadas o “*master node*” faz um *merge* dos arquivos e que são então destinados para as tarefas de *reduce*.

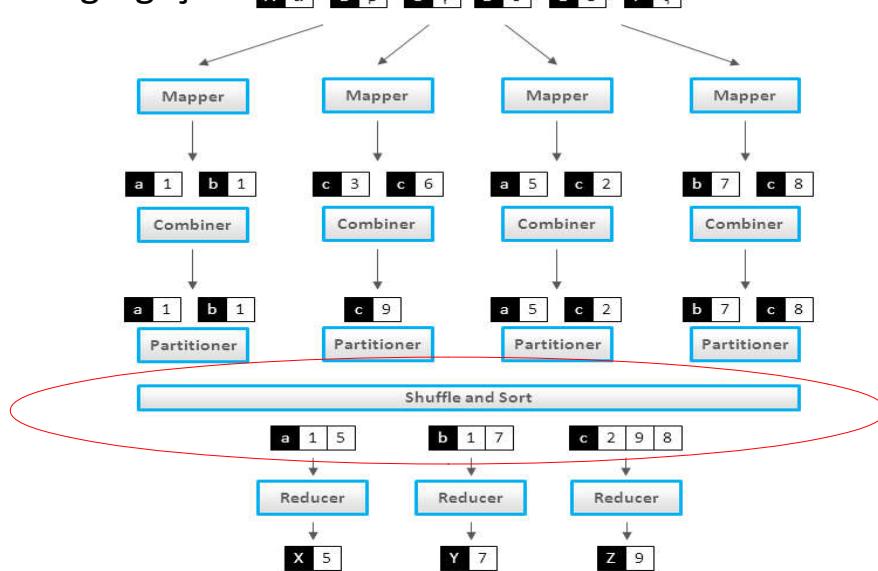
Claudio Lúcio

25

Algoritmo Map Reduce

Detalhes de funcionamento:

Agrupamento e agregação:



Algoritmo Map Reduce

Detalhes de funcionamento:

Tarefas *Reduce*:

- Para cada chave k , a tarefa *reduce* recebe um conjunto de pares na forma $(k, [v_1, v_2, \dots, v_n])$ oriundos de várias tarefas *map* na forma $(k, v_1)(k, v_2) \dots (k, v_n)$;
- A tarefa *reduce* deve combinar os valores de alguma forma;

Cláudio Lúcio

27

Algoritmo Map Reduce

Detalhes de funcionamento:

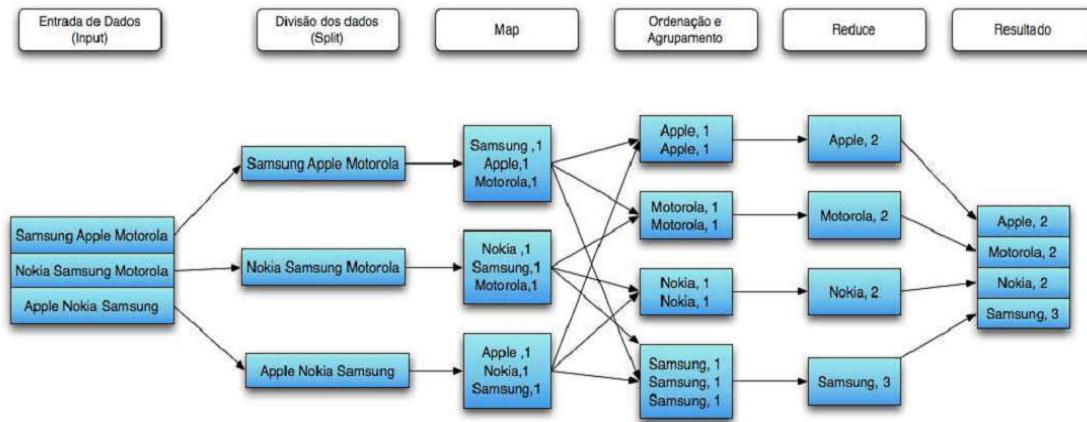
Tarefas *Reduce*:

- Exemplo clássico - Contar o número de palavras em um coleção de documentos:
 - A função *reduce* apenas faz a soma dos valores para cada uma das chaves w_h
 - Desta forma a saída da função *reduce* será um conjunto de chaves-valor na forma (w, m) ;
 - Em que m é o total de vezes que a palavra w aparece em todos os documentos;

Algoritmo Map Reduce

Detalhes de funcionamento:

Exemplo clássico:



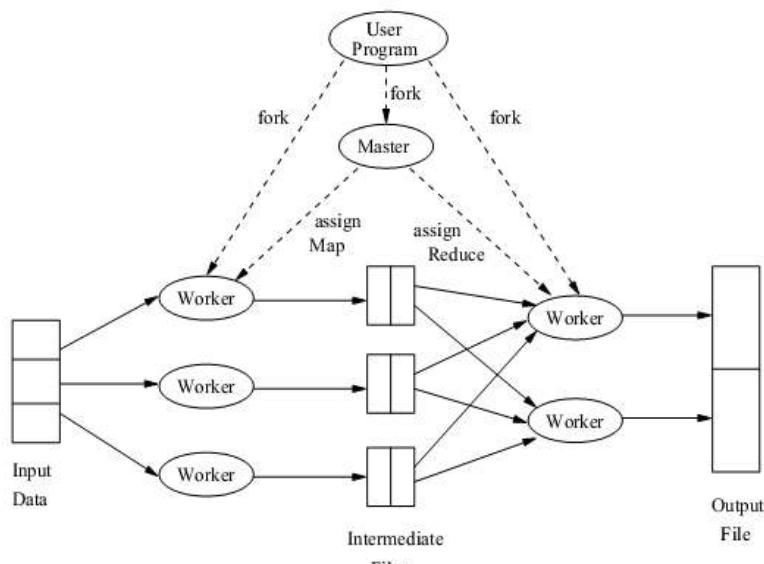
Cláudio Lúcio

29

Algoritmo Map Reduce

Detalhes de funcionamento:

Outras considerações: processos, arquivos e tarefas



Algoritmo Map Reduce

Detalhes de funcionamento:

Outras considerações:

- Revisão: processos, arquivos e tarefas
 - O '*Name node*' é responsável pela criação das tarefas de Map e Reduce nos '*data nodes*' ;
 - É ideal que se crie um tarefa *map* para cada bloco;
 - O número de tarefas *reduce* deve ser mais reduzido, a intenção é evitar a explosão do número de arquivos gerados pela tarefa *map* (um arquivo para cada *reduce*);
 - O '*name node*' acompanha a execução das tarefas (em espera, executando, completo); Quando um processo termina ele comunica para o '*name node*';

Cláudio Lúcio

31

Algoritmo Map Reduce

Detalhes de funcionamento:

Outras considerações:

- Revisão: processos, arquivos e tarefas
 - Cada tarefa *map* processa vários blocos de vários arquivos;
 - Os arquivos gerados pelo *map* são gravados localmente e o '*name node*' armazena todos estes dados: nomes dos arquivos e tamanho;
 - Uma tarefa *reduce* recebe todos os arquivos intermediários para o processamento da função *reduce*;

Algoritmo Map Reduce

Resumo Python:

Variáveis e Substituição:

```
lista = [ 1, 2, "texto", 3.5 ]
print lista[ 0 ] # imprime 1
print lista[ 1 : 2 ] # imprime [ 2, "texto" ]
print lista[ -1 ] # imprime [ 1, 2, "texto" ]
lista += [ "novo" ]
print lista # imprime [ 1, 2, "texto", 3.5, "novo" ]

tupla = ( 1, 2, "texto", 3.5 ) # Elementos não podem ser alterados!
print tupla[ 0 ] # imprime 1
print tupla[ 1 : 2 ] # imprime ( 2, "texto" )
print tupla[ : -1 ] # imprime ( 1, 2, "texto" )
tupla += ( "novo", )
print tupla # imprime ( 1, 2, "texto", 3.5, "novo" )

dicionario = { "chave": "valor", "c2": "v2" }
print dicionario[ "chave" ] # imprime valor

newstring1 = "string"%s int=%d float=%03.2f" % ( "txt", 12, 4.56 )
newstring2 = "chave=%(chave)s c2=%(c2)s" % dicionario
newstring3 = "chave=%s c2=%s" % ( dicionario[ "chave" ],
                                 dicionario[ "c2" ] )
```

Controle de Fluxo e Laços:

```
if a > b and a < c:
    print "entre b e c"
elif a > c:
    print "a maior que c"
else:
    print "a menor ou igual a b ou igual a c"

for elemento in lista:
    print "elemento: %s" % elemento

coordenadas = [ ( 0, 0, 0 ), ( 1, 0, 0 ), ( 0, 1, 0 ), ( 0, 0, 1 ) ]
for x, y, z in coordenadas:
    print "Ponto: x=%d, y=%d, z=%d" % ( x, y, z )

loop = 1
while loop:
    resultado = faca_acao()
    if resultado < 0:
        break # Para o laço
    else resultado > 0:
        continua # Volta para o começo do laço
print "teste"
```

Funções:

```
def funcao( p1, p2="Valor Padrão" ):
    print "p1: %s" % p1, "p2: %s" % ( p1, p2 )

def f_param_variaveis( p1, *args ):
    print "p1: %s" % p1
    for arg in args:
        print "arg: %s" % arg

def f_param_nome_variaveis( p1, **args ):
    print "p1: %s" % p1
    for p_name, p_value in args:
        print "arg: %s=%s" % ( p_name, p_value )
```

Classes:

```
class A:
    atributo = 1
    __privado = 123
    def __init__( self, valor ):
        self.atributo = valor
        self.__metodo_privado()
    def __metodo_privado( self ):
        print "chamando metodo privado"

class B:
    atributo = 2
    def __init__( self ):
        self.novo_atributo = 2
class C( A, B ):
    def __init__( self ):
        B.__init__( self )
class D( B, A ):
    def __init__( self ):
        B.__init__( self )
a = A( 1 )
b = B()
c = C()
d = D()

print a.atributo # imprime 1
print b.atributo # imprime 2
print c.atributo # imprime 1: herança múlt. (A,B) A sobrepuja-se a B
print d.atributo # imprime 2: herança múlt. (B,A) B sobrepuja-se a A
```

Módulos e Espaço de Nomes:

```
import urllib
url = "http://www.unicamp.br/" + urllib.quote( "index.html" )
conteudo = urlopen( url ).read()

# importa símbolos para espaço de nomes atual:
from urllib import *
url = "http://www.unicamp.br/" + quote( "index.html" )
conteudo = urlopen( url ).read()
```

33

Algoritmo Map Reduce

Exemplo - Mincemeat:

```
#!/usr/bin/env python
import mincemeat

data = ["Humpty Dumpty sat on a wall",
        "Humpty Dumpty had a great fall",
        "All the King's horses and all the King's men",
        "Couldn't put Humpty together again",
        ]

def mapfn(k, v):
    for w in v.split():
        yield w, 1

def reducefn(k, vs):
    result = 0
    for v in vs:
        result += v
    return result

s = mincemeat.Server()

# The data source can be any dictionary-like object
s.datasource = dict(enumerate(data))
s.mapfn = mapfn
s.reducefn = reducefn

results = s.run_server(password="changeme")
print results
```

Algoritmo Map Reduce

Exemplo – Mincemeat:

- Servidor – '*Name node*':

```
python example.py
```

- Clientes – '*data node*':

```
python mincemeat.py -p changeme [server address]
```

- Resultado:

```
{'a': 2, 'on': 1, 'great': 1, 'Humpty': 3, 'again': 1, 'wall': 1, 'Dumpty': 2, 'men': 1, 'had': 1, 'all': 1, 'together':
```

Cláudio Lúcio

35

Algoritmo Map Reduce

Tipo de processamentos:

- *Map reduce* não é uma solução para qualquer tipo de problema. Ex. de exceção: site de comércio eletrônico;
- Pode ser utilizado:
 - Encontrar palavras chaves;
 - Operações que envolvam multiplicação de matrizes;
 - Operações de álgebra relacional:
 - Seleção - σ
 - Projeção - π
 - União, interseção e diferença
 - Natural *join* - \bowtie
 - Agrupamentos e agregações

Algoritmo Map Reduce

Natural Join – SQL:

- Suponha a duas relações $R(A,B)$ e $S(B,C)$

Função *Map*:

- Para cada tupla a, b de R produza um membro chave valor $(b, (R, a))$;
 - Para cada tupla de b, c de S produza um membro chave valor $(b, (S, c))$;
- Função *Reduce*:
 - Cada chave b deve ser associada com ambos os itens (R, a) e (S, c) ;
 - A saída para a chave b deve ser $(b, [(a_1, b, c_1), (a_2, b, c_2), \dots])$

Cláudio Lúcio

37

Algoritmo Map Reduce

- Exercícios práticos
 - Exercício de contagem de palavras para textos
 - Exercício de implementação de natural join e agrupamentos;
- Trabalho prático
 - Palavras chaves por autores;



Um conjunto de comentários sobre os produtos da sua empresa foi disponibilizado (textos em inglês). Todos eles são comentários positivos sobre os produtos. O seu gerente solicitou que você prepare uma solução que possa atender os seguinte requisitos:

- I. Armazenar os textos e extrair 15 palavras chaves que são mais citadas pelos clientes;
- II. Sua companhia possui 4 milhões de clientes e há uma interação com estes clientes pelo menos uma vez por mês;
- III. Para cada mês deve haver uma atualização nas palavras ;
- IV. Em média um cliente interage com a companhia 1 vez por mês; Cada interação gera um arquivo texto a partir do sistema fonte de informação que armazena este dado;
- V. Algumas palavras deve ser descartadas: 'by', 'above', 'all', 'none', 'nobody'.....

Sua missão é fazer uma prova de conceito para o seu gerente. Ele ouviu falar de processamento paralelo e leu um artigo na '*Computer magazine*' falando sobre o movimento NoSQL: quer entender no cenário da empresa como isto poderia funcionar. Ele já solicitou um extração de 1000 arquivos sobre os comentário dos clientes. Em uma reunião foi definido que você deve entregar um exemplo funcionando amanhã

Você deve então:

- a- Usar uma implementação simples e rápida;
- b- Permitir que isto seja executada em 50 máquinas na empresa;
- c- No final gerar um arquivo com a contagem das palavras;

Vamos começar com a implementação:

1. Faça a instalação do python 2.7;
2. Crie o diretório exerc\ ;
3. Crie o diretório exerc\textos;
4. Copie o arquivo zipado(2.1 - textos.zip) para o diretório criado no passo anterior;
5. Copie o arquivo python do mincemeat para o diretório exerc\
<https://github.com/michaelfairley/mincemeatpy>
6. Crie um arquivo texto e digite a primeira parte do código – Importação dos arquivos:

```
import mincemeat
import glob
import csv

text_files = glob.glob('Exerc\\textos\\*')

def file_contents(file_name):
    f = open(file_name)
    try:
        return f.read()
    finally:
        f.close()

source = dict((file_name, file_contents(file_name))for file_name in text_files)
```

Estas parte faz a importação dos dados e gera um objeto 'source' que é do tipo dicionário: nome do arquivo e conteúdo do arquivo;

7. Faça a implementação do método *map*:

```

def mapfn(k, v):
    print 'map ' + k
    from stopwords import allstopwords
    for line in v.splitlines():
        for word in line.split():
            if (word not in allstopwords):
                yield word, 1

```

Neste caso esta sendo utilizada um implementação rápida e simples;

8. Faça a implementação do método *reduce*:

```

def reducefn(k, v):
    print 'reduce ' + k
    return sum(v)

```

9. Utize agora o *mincemeat* ele vai simular o papel da DFS e do '*name mode*':

```

s = mincemeat.Server()
# A fonte de dados pode ser qualquer objeto do tipo dicionário
s.datasource = source
s.mapfn = mapfn
s.reducefn = reducefn
results = s.run_server(password="changeme")
w = csv.writer(open("Exerc\\RESULT.csv", "w"))
for k, v in results.items():
    w.writerow([k, v])

```

10. Salve o nome do arquivo como exerc21.py;

11. Vamos fazer a execução paralela deste código:

a- server: python exerc21.py

b- Crie dois ou três clientes com o comando:

python mincemeat.py -p changeme localhost

Para executar remoto,faça a instalação do python e mincemeat conforme passos 1 e 4, respectivamente;

12. Veja o arquivo gerado



Neste caso vamos implementar um junção e agrupamento de duas tabelas que estão representadas por dois arquivos:

I. Vendas:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Store 01	Customer 144	80759 Books		29/11/2006 06:48	9	80,5	1												
2	Store 04	Customer 1716	56810 Clothing		14/01/2007 21:15	5	42,3	2												
3	Store 04	Customer 129	60636 Movies		17/12/2005 13:26	5	92,9	3												
4	Store 10	Customer 802	82105 Home/Garden		06/03/2007 19:57	4	70,6	4												
5	Store 14	Customer 1	52349 Clothing		09/12/2005 21:52	8	76,6	5												
6	Store 07	Customer 1190	43024 Movies		31/05/2005 11:18	7	90,2	6												
7	Store 14	Customer 1268	82054 Movies		08/02/2006 20:58	1	26,1	7												
8	Store 10	Customer 1433	57181 Electronics		19/06/2007 18:06	9	66,4	8												
9	Store 05	Customer 103	15211 Health		11/02/2005 18:14	9	83,5	9												
10	Store 01	Customer 1871	72478 Electronics		24/06/2006 10:10	6	77,4	10												
11	Store 07	Customer 1193	50965 Movies		14/09/2006 02:40	2	69,8	11												
12	Store 07	Customer 292	97284 Movies		01/07/2008 16:59	7	42,7	12												
13	Store 14	Customer 751	30215 Home/Garden		04/09/2005 15:43	6	39,3	13												
14	Store 13	Customer 1736	96441 Sports		27/11/2006 21:44	9	89,8	14												
15	Store 13	Customer 1740	51946 Sports		16/02/2005 10:18	4	41,2	15												
16	Store 14	Customer 307	96842 Clothing		18/01/2008 16:03	3	14,9	16												
17	Store 09	Customer 30	89799 Clothing		21/03/2007 13:17	1	81,2	17												
18	Store 15	Customer 1837	84656 Movies		04/11/2005 06:00	8	76,3	18												
19	Store 02	Customer 1641	62589 Toys		11/09/2006 22:59	4	12,7	19												
20	Store 03	Customer 191	18438 Electronics		16/03/2008 22:40	9	14,2	20												
21	Store 13	Customer 1289	53573 Sports		07/11/2008 03:51	8	37,1	21												
22	Store 10	Customer 1575	50955 Home/Garden		13/01/2008 05:38	9	27	22												
23	Store 09	Customer 484	92160 Home/Garden		14/06/2007 19:50	5	63,3	23												

II. Filiais:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Store 01	CO_BHS																		
2	Store 04	CO_POA																		
3	Store 10	CO_NAS																		
4	Store 14	CO_CPT																		
5	Store 07	CO_NHS																		
6	Store 05	CO_BET																		
7	Store 13	NA_GFT																		
8	Store 09	GH_DSE																		
9	Store 15	DF_FGE																		
10	Store 03	QW_ERT																		
11	Store 03	MN_DFG																		
12	Store 12	YU_GHJ																		
13	Store 11	EW_MKL																		
14	Store 08	TR_RTE																		
15	Store 06	PO_FGD																		

Neste caso o que se deseja é fazer um junção dos dois arquivos para que seja apresentado um resultado que seria: Código da filial (arquivo Filial - campo1), descrição da filial (arquivo Filial - campo2) e total de itens pedidos (arquivo Filial – campo6);
Na verdade seria um SQL com join e um group by:

```
Select cod_filial, des_filial, sum(qtd_item) as total
from vendas inner join filial on(filial.cod_filial = vendas.cod_filial)
group by cod_filial, des_filial
```

Implementação:

1. Faça a instalação do python 2.7;
2. Crie o diretório exerc\;
3. Crie o diretório exerc\join;
4. Copie o arquivo zipado(2.2 Join.zip) para o diretório criado no passo anterior;
5. Copie o arquivo python do mincemeat para o diretório exerc\
<https://github.com/michaelfairley/mincemeatpy>
6. Crie um arquivo texto e digite a primeira parte do código – Importação dos arquivos:

```
import mincemeat
import glob
import csv

text_files = glob.glob('G:\\NOSQL\\Exerc\\Join\\*')

def file_contents(file_name):
    f = open(file_name)
    try:
        return f.read()
    finally:
        f.close()

source = dict((file_name, file_contents(file_name))for file_name in text_files)
```

Estas parte faz a importação dos dados e gera um objeto 'source' que é do tipo dicionário: nome do arquivo e conteúdo do arquivo;

7. Faça a implementação do método *map*:

```
def mapfn(k, v):
    print 'map ' + k
    for line in v.splitlines():
        if k == 'G:\\NOSQL\\Exercícios\\1.3 Join\\1.3-vendas.csv':
            yield line.split(';')[0], 'vendas' + ';' + line.split(';')[5]
        if k == 'G:\\NOSQL\\Exercícios\\1.3 Join\\1.3-filiais.csv':
            yield line.split(';')[0], 'Filial' + ';' + line.split(';')[1]
```

Altera os 'if' acima para o caminho e nomes dos arquivos da sua estação de trabalho;

8. Faça a implementação do método *reduce*. Implementação simples para entender o que acontece:

```
def reducefn(k, v):
    print 'reduce ' + k
    return v
```

9. Utize agora o *mincemeat* ele vai simular o papel da DFS e do '*name mode*':

```
s = mincemeat.Server()

# A fonte de dados pode ser qualquer objeto do tipo dicionário
s.datasource = source
s.mapfn = mapfn
s.reducefn = reducefn

results = s.run_server(password="changeme")

w = csv.writer(open("Exerc\\RESULT.csv", "w"))
for k, v in results.items():
    w.writerow([k, v])
```

10. Salve o nome do arquivo como exerc22.py;

11. Faça a execução paralela deste código:

a- server: python exerc22.py

b- Crie dois ou três clientes com o comando:

python mincemeat.py -p changeme localhost

Para executar remoto,faça a instalação do python e mincemeat conforme

passos 1 e 4, respectivamente;

12. Veja o arquivo gerado

13. Faça a implementação final do método *reduce*:

```
def reducefn(k, v):
    print 'reduce' + k
    total = 0
    for index, item in enumerate(v):
        if item.split(":")[0] == 'vendas':
            total = int(item.split(":")[1]) +total
        if item.split(":")[0] == 'Filial':
            NomeFilial = item.split(":")[1]
    L = list()
    L.append(NomeFilial + " , " + str(total))
    return L
```

14. Utize agora o *mincemeat* novamente:

```
s = mincemeat.Server()
s.datasource = source
s.mapfn = mapfn
s.reducefn = reducefn
results = s.run_server(password="changeme")
w = csv.writer(open("G:\\NOSQL\\Exercicios\\RESULT_1.3.csv", "w"))
for k, v in results.items():
    w.writerow([k, str(v).replace("[", "").replace("]", "").replace("'", "")])
```

15. Faça a execução paralela deste código:

a- server: python exerc22.py

b- Crie dois ou três clientes com o comando:

python mincemeat.py -p changeme localhost

Para executar remoto, faça a instalação do python e mincemeat conforme passos 1 e 4, respectivamente;

16. Veja o arquivo gerado



Faça o download dos arquivos Trab2.3.zip

Cada arquivo neste diretório possui a seguinte estrutura:

journals/cl/SantoNR90:::Michele Di Santo:::Libero Nigro:::Wilma
Russo:::Programmer- Defined Control Abstractions in Modula-2.

Cada uma das linhas pode ser entendida como:

- I. Informação bibliográfica;
- II. autores separados por ':::';
- III. Título da obra

paper-id:::author1:::author2:::.... ::authorN:::title

A tarefa é fazer o cálculo de quantas vezes cada termo (no título da obra) acontece por autor;

Por exemplo para o autor 'Alberto Pettorossi' os seguintes termos acontecem (considerando todos os documentos): program:3, transformation:2, transforming:2, using:2, programs:2, and logic:2.

Observe os seguintes itens:

1. O separador de campos é “:::” e separador de autores é “:::”;
2. Cada autor pode ter escrito múltiplas obras, que por sua vez podem estar em vários arquivos;
3. Existe uma lista de palavras que não serão consideradas. Utilize o arquivo `stopword.py` do exercício prático: exclua todas estas palavras para os autores;
4. Se possível faça a exclusão de pontuações também, de forma que a palavra `logic` e `logic.` sejam tratadas como se fossem uma única palavra;
5. Entregue a implementação do seu grupo;
6. Responda quais são as duas palavras que mais acontecem para os seguintes autores:
 - a- “Grzegorz Rozenberg”
 - b- “Philip S. Yu”



Paradigmas de computação distribuída e bancos de dados

Consistência em bancos de dados não relacionais

Cláudio Lúcio

39

Consistência em bancos de dados não relacionais

- Consistência estrita
 - Todas as operações de leitura devem retornar dados da última operação de escrita completa;
 - Operações de leitura e escrita em um único nó;
 - Utilização de protocolos de transação distribuída;
 - De acordo com o teorema CAP, não pode ser conseguida em conjunto com disponibilidade e tolerância a partição;

Consistência em bancos de dados não relacionais

- Consistência eventual
 - As operações de leitura irão, eventualmente, ler dados da última operação de escrita;
 - Pode haver leituras inconsistentes pois o sistema geral esta em atualização;
 - Em um *cluster* uma leitura pode ser feita a partir de uma réplica que ainda não foi atualizada, pois o último processo de escrita aconteceu em outra réplica do dado;
 - Em alguns sistemas este tempo de atualização das réplicas pode ser 500 milisegundos;

Cláudio Lúcio

41

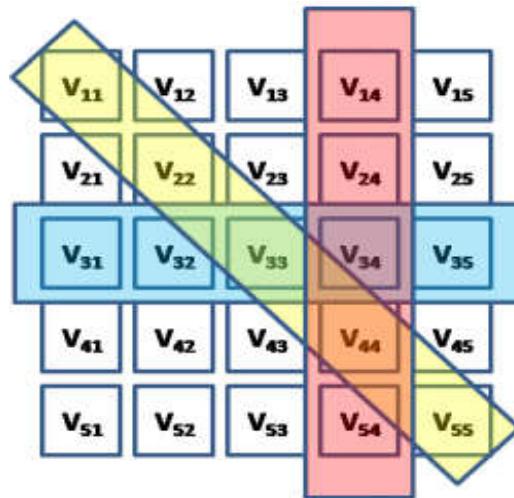
Consistência em bancos de dados não relacionais

- Consistência eventual
 - Se bancos de dados são distribuídos (vários nós), há a chance de dados serem lidos e alterados em qualquer nó;
 - Consistência estrita não é possível neste cenário;
 - Desta forma existe a necessidade de controlar modificações e versões concorrentes para quais o estado final do banco de dados vai convergir;
- Tratamento e versionamento: ambientes distribuídos
 - Timestamp: exige a necessidade de um relógio sincronizado em todo *cluster*;
 - Vetores de relógios: muito usado, mas exige alguma forma de resolver conflitos;

Consistência em bancos de dados não relacionais

- Vetores de relógios

- Item i da cópia local – **diagonal**;
- O que a réplica sabe sobre a atualização das demais réplicas – **linhas** da matriz;
- O que todos sabem sobre uma determinada réplica – **colunas** da matriz ;

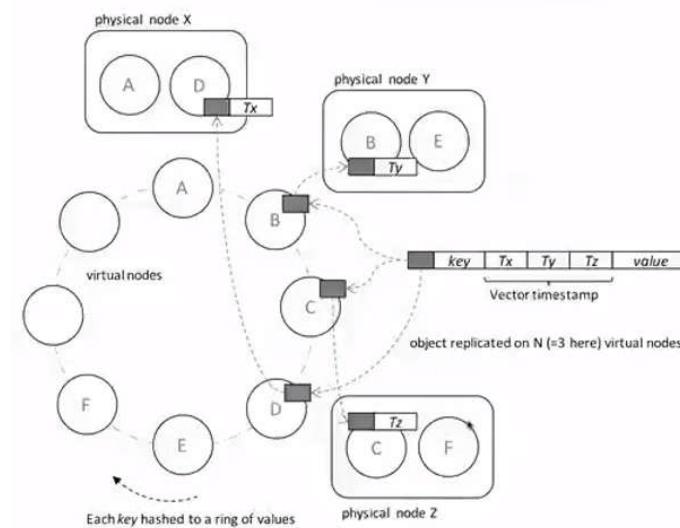


Cláudio Lúcio

43

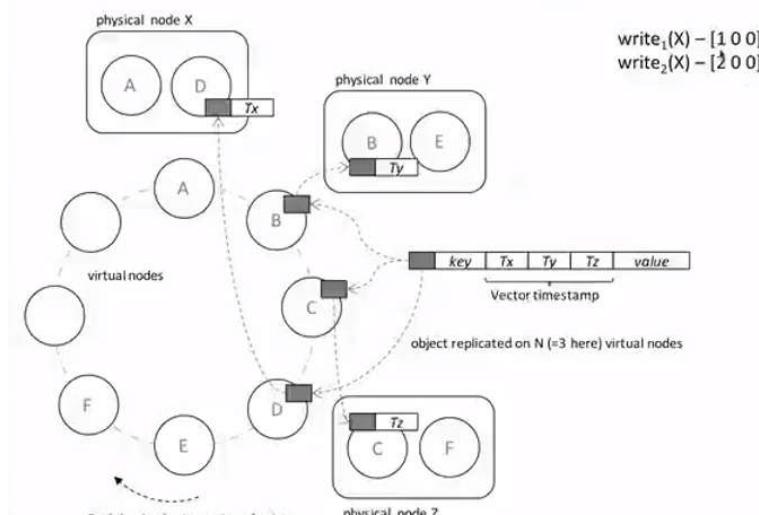
Consistência em bancos de dados não relacionais

- Vetores de relógios



Consistência em bancos de dados não relacionais

- Vetores de relógios



Cláudio Lúcio

45

NoSQL

MongoDB

MapReduce

Cassandra

Python

Processamento

BerkeleyDB

Paralelismo

DFS

chave-Valor

Documentos

Colunar

Bancos de dados baseados em chave valor

Cláudio Lúcio

1

Bancos de dados Chave-Valor

Agenda

- Conceitos
- Amazon Dynamo DB



Bancos de dados baseados em chave valor

Conceitos

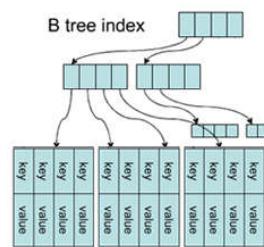
Cláudio Lúcio

3

Conceitos

Proposta Chave Valor:

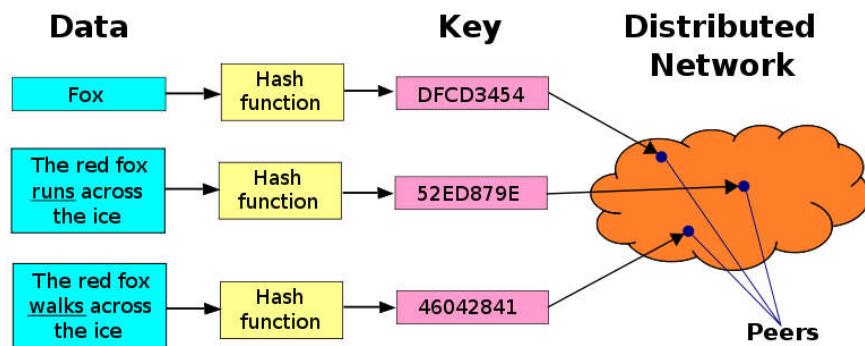
- Modelos de dados simples;
- Em essência;
 - Estrutura de mapas (*hash maps*) ou dicionários(chave-valor);
 - Permitir os clientes: *put(chave,valor)* e *get/request(chave)*;
- Boa parte das soluções chave valor favorecem a escalabilidade (versus consistência);



Conceitos

Proposta Chave Valor:

- Origem: Distributed hash table - DHT;



Cláudio Lúcio

5

Conceitos

Proposta Chave Valor:

- Ausência de algumas funções analíticas: junções e agregações;
- Armazenamento das chaves é limitado (bytes) e normalmente não há restrições para os valores;
- Pode ser dizer 'ausência de esquema':
 - Chave valor;
 - Algumas soluções, REDIS, transforma o valor em listas, grupos ou *hash*;

Conceitos

Algumas soluções:

- Redis:
 - Site: <http://redis.io/>;
 - Histórico: Iniciado em 2009 por Salvatore Sanfilippo em sua *startup* (<http://lloogg.com/>). Vmware patrocina o projeto;
 - Linguagem: foi implementado em C;
 - Métodos de acesso: conjunto de métodos e operações que podem ser acessado através de linha de comando ou utilizando bibliotecas de linguagens como: Java, Python e Ruby;

Cláudio Lúcio

7

Conceitos

Algumas soluções:

- Riak:
 - Site: <http://wiki.basho.com>;
 - Histórico: Criado pela empresa Basho (empresa criada em 2008) ;
 - Linguagem: foi implementado em Erlang, faz uso também de C e JavaScript;
 - Métodos de acesso: possui interface para JSON (sobre HTTP). Existem bibliotecas para Erlang, Java, Ruby, Python, PHP e JavaScript;

Cláudio Lúcio

Conceitos

Algumas soluções:

- Voldemort:
 - Site: <http://project-voldemort.com>;
 - Histórico: Criado pelo time de dados e *analytics* do Linkedin;
 - Linguagem: foi implementado em Java;
 - Métodos de acesso: JAVA e pode ser usado com HADOOP;
 - Linkedin faz uso desta solução;

Cláudio Lúcio

9



Amazon Dynamo DB

Características:

- Arquitetura distribuída - '*shared nothing*';
- Orientado para desempenho:
 - Na criação da tabela deve se especificar as características de desempenho (leitura/escrita);
 - Automaticamente aloca os recursos para a tabela e faz a partição dos dados de acordo com o número de servidores disponíveis e desempenho especificado;
- Não existe limite(volume) para uma tabela;
- Escalabilidade horizontal: centenas de servidores;

Cláudio Lúcio

11

Amazon Dynamo DB

Características:

- API simplificada - itens:
 - *Get, Put, Delete e Update;*
 - API disponíveis em .NET, PHP e Java;
- Operações com tabelas
 - Permite a criação, atualização e exclusão de tabelas;
 - A atualização pode alterar aspectos relativos ao desempenho;
 - Função de descrição da tabela (metadados) - *describe*;
 - Possui operações de *query* e *scan table*;

Cláudio Lúcio

Amazon Dynamo DB

Características:

- Segurança
 - Provê mecanismos de autenticação e autorização;
 - Mecanismo de segurança integrado com outros produtos Amazon;
- Administração: interface simples para gerenciamento e manutenção - *AWS Management Console*;
- Possui integração com o Amazon EMR - Amazon Elastic MapReduce:
 - HADOOP no *cloud Amazon*;
- Também com outros produtos Amazon ;

Cláudio Lúcio

13

Amazon Dynamo DB

Modelos de dados:

- Inclui tabelas, itens e atributos;
- Banco de dados: coleção de tabelas;
- Tabela: coleção de itens;
 - Toda tabela deve possuir uma chave primária;
- Item: coleção de atributos;
 - Não há limite no número de atributos;
 - Existe um limite de 64 KB no tamanho do item (soma do tamanho dos atributos);

Amazon Dynamo DB

Modelos de dados:

- Atributos
 - Cada atributo é um par: Nome e valor ;
 - Atributos podem ser multi valorados;
 - Tipos de dados
 - Escalares: *Number, String e Binary*;
 - Multi-valorados: *String Set, Number Set e Binary Set*.
 - Não permite campos nulos ou vazios (*strings*);

Cláudio Lúcio

15

Amazon Dynamo DB

Modelos de dados:

- Ausência de esquema;

```
{  
    Id = 101  
    ProductName = "Book 101 Title"  
    ISBN = "111-1111111111"  
    Authors = [ "Author 1", "Author 2" ]  
    Price = -2  
    Dimensions = "8.5 x 11.0 x 0.5"  
    PageCount = 500  
    InPublication = 1  
    ProductCategory = "Book"  
}
```

```
{  
    Id = 201  
    ProductName = "18-Bicycle 201"  
    Description = "201 description"  
    BicycleType = "Road"  
    Brand = "Brand-Company A"  
    Price = 100  
    Gender = "M"  
    Color = [ "Red", "Black" ]  
    ProductCategory = "Bike"  
}
```

```
{  
    Id = 202  
    ProductName = "21-Bicycle 202"  
    Description = "202 description"  
    BicycleType = "Road"  
    Brand = "Brand-Company A"  
    Price = 200  
    Gender = "M"  
    Color = [ "Green", "Black" ]  
    ProductCategory = "Bike"  
}
```

Amazon Dynamo DB

Modelos de dados:

- Chave Primárias
 - Tipo *hash* : índice hash não ordenado
 - Tipo *hash* e intervalar:
 - Dois atributos;
 - O primeiro é o *hash*
 - O segundo é o intervalar:índice ordenado no *hash*;

Table Name	Primary Key Type	Hash Attribute Name	Range Attribute Name
Forum (<u>Name</u> , ...)	Hash	Attribute Name: Name	-
Thread (ForumName, <u>Subject</u> , ...)	Hash and Range	Attribute Name: ForumName	Attribute Name: Subject
Reply (<u>Id</u> , ReplyDateTime, ...)	Hash and Range	Attribute Name: Id	Attribute Name: ReplyDateTime

Cláudio Lúcio

17

Amazon Dynamo DB

Modelos de dados:

- No site da Amazon existem algumas recomendações de 'melhores práticas' para utilização:
 - Utilizar tabela '1 para muitos' ao invés de grandes conjuntos de atributos:

ProductCatalog

<u>Id</u>	Title	Description
21	"Famous Book"	"From last year's best seller's list, ..."
302	"Red Bicycle"	"Classic red bicycle..."
58	"Music Album"	"A new popular album from this week..."

ProductAvailability

<u>Id</u>	Price	InStockCount
21	"\$5.00 USD"	3750
302	"\$125.00 USD"	8
58	"\$5.00 USD"	"infinity (digital item)"

Amazon Dynamo DB

Modelos de dados:

- No site da Amazon existem algumas recomendações de 'melhores práticas' para utilização:
 - Compactação de grandes conjuntos de atributos;
 - Partição de grandes atributos em múltiplos itens:

Reply

<u>Id</u>	<u>ReplyDateTime</u>	Message	ChunkCount	ChunkVersion
"Amazon DynamoDB#Thread1"	"2012-03-15T20:42:54.023Z"		3	1
"Amazon DynamoDB#Thread2"	"2012-03-21T20:41:23.192Z"	"short message"		

ReplyChunks

<u>Id</u>	Message
"Amazon DynamoDB#Thread1#2012-03-15T20:42:54.023Z#1#1"	"first part of long message text..."
"Amazon DynamoDB#Thread1#2012-03-15T20:42:54.023Z#1#3"	"...third part of long message text"
"Amazon DynamoDB#Thread1#2012-03-15T20:42:54.023Z#1#2"	"...second part of long message text..."

Cláudio Lúcio

19

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- Não há diferenças entre os nós do *cluster*, todos tem o mesmo papel;
- Suporta nós com diferentes configurações de hardware: faz balanceamento de carga considerando estes quesitos;
- Uma requisição de cliente pode ser atendida por qualquer nó do *cluster*;
 - Um processo fica responsável por fazer a indicação de 'roteamento' chave – nós do *cluster*;
 - Biblioteca cliente determina o particionamento das chaves nos nós do *cluster*;

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- Utiliza o algoritmo '*consistent hashing*' para particionar chaves para nós do *cluster*;
- Um *hash map* é criado para as chaves e nós do cluster;
- Em um sentido horário:
 - Os intervalos das funções *hash* são distribuídos para um nó;
 - Cada nó também faz uso de seu vizinho para repassar réplicas;
 - De forma que se ele deixa o *cluster* o vizinho tem condição de assumir a chave;
- Neste algoritmo os próprios clientes podem usar a função *hash* para determinar em quais nós a chave deve residir;

Cláudio Lúcio

21

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- '*Consistent hashing*' – Exemplo:

'Hashing' comum

<u>Chaves</u>	<u>Servidores</u>
K0=380	
k1=983	1 2 3
K2=765	
...	
Kn=987	64

Hash table

K0=Servidor 1
k1=Servidor 3
K2=Servidor 10
...
Kn=Servidor 50

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- '*Consistent hashing*' – Exemplo:

'Hashing' comum

<u>Chaves</u>	<u>Servidores</u>
K0=380	
k1=983	1 2 3
K2=765	
...	
Kn=987	64
<u>Hash table</u>	
	K0=Servidor 1
	k1=Servidor 3
	K2=Servidor 10
	...
	Kn=Servidor 50

O que acontece se o
número de servidores
aumenta ??

Cláudio Lúcio

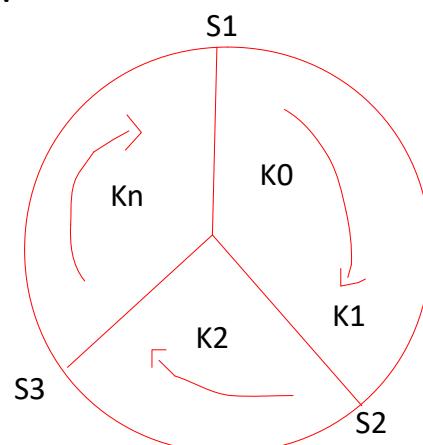
23

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- '*Consistent hashing*' – Exemplo:

<u>Chaves</u>	<u>Servidores</u>
K0=380	S1
K1=983	S2
K2=765	S3
...	
Kn=987	



Cláudio Lúcio

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

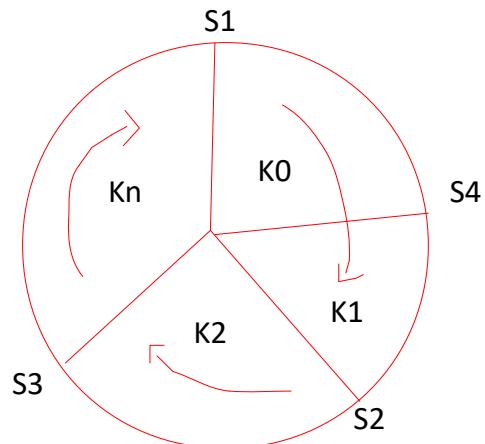
- '*Consistent hashing*' – Exemplo:

Chaves

K0=380
K1=983
K2=765
...
Kn=987

Servidores

S1
S2
S3
S4



Cláudio Lúcio

25

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

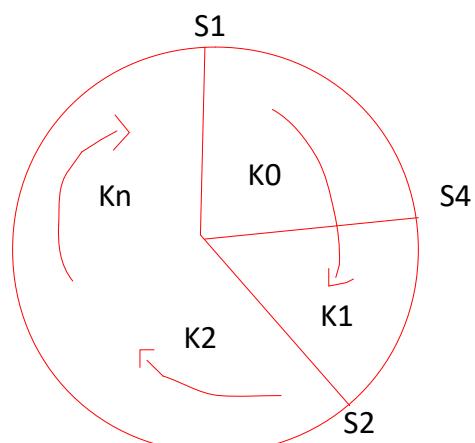
- '*Consistent hashing*' – Exemplo:

Chaves

K0=380
K1=983
K2=765
...
Kn=987

Servidores

S1
S2
S4



Cláudio Lúcio

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- '*Consistent hashing*'
 - Dada uma solicitação para encontrar a chave;
 - Todos os servidores mantém uma estrutura de distribuição das chaves (*hash map*);
 - A requisição pode ser atendida por qualquer servidor;
 - Se ele possuir o dado, a requisição será atendida imediatamente, caso contrário ele já sabe qual servidor possui a chave e encaminha a requisição;

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- '*Consistent hashing*'
 - No Dynamo houve alteração no algoritmo original;
 - Original:
 - A distribuição dos nós no anel é aleatória(posição);
 - Esta distribuição para a função *hash* pode causar um desbalanceamento no caso de máquinas heterogêneas;
 - Dynamo:
 - Utiliza o conceito de *virtual node*;
 - Máquinas com mais recursos possuem mais *virtual nodes* e vice versa;

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- Réplicas
 - Se um nó deixa o *cluster*, os dados armazenados ficam indisponíveis, caso não tenham sido replicados;
 - No caso de um novo nó no *cluster*, nós adjacentes deixam de ser responsáveis por uma porção dos dados, mas ainda retém os dados;
 - Existe fator de replicação - r : os próximos r nós adjacentes (sentido horário) devem reter réplicas dos dados e serem também responsáveis pelos dados (lista de nós);

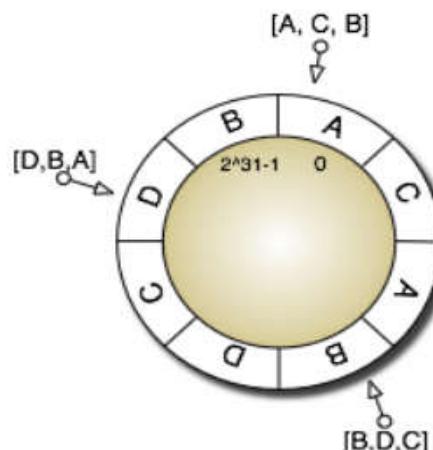
Cláudio Lúcio

29

Amazon Dynamo DB

Escalabilidade(tolerância a partição):

- Réplicas
 - Exemplo com fator de replicação $r=3$:



Amazon Dynamo DB

Disponibilidade:

- Possui um processo para descobrir nós que não estão mais ativos no *cluster*;
- No caso de uma atualização em nó preferencial:
 - Se o nó não está disponível;
 - Processo coordenador indica outro nó (auxiliar) e faz uma réplica temporária dos dados com a atualização;
 - Assim que o nó retorna ao *cluster* a atualização com base na réplica temporária é realizada;
 - Réplica temporária é excluída do nó auxiliar;

Amazon Dynamo DB

Disponibilidade:

- O processo de adição e remoção de nós no cluster é feito por um administrador;
- Na adição ou remoção o '*gossip protocol*' é utilizado para repassar a informação da exclusão/inclusão do nó;
- A adição de nós no *cluster* gera a alteração de faixas de chaves no anel do '*consistent hash*' ;
 - Quando os nós descobrem que um nó foi adicionado, há a determinação de chaves que não estão mais sobre seu controle;

Amazon Dynamo DB

Disponibilidade:

- A adição de nós no *cluster* gera a alteração de faixas de chaves no anel do '*consistent hash*' ;
 - A transferências das chaves é realizada;
 - De acordo com o caso, as chaves são excluídas do nó antigo;
- Quando um nó é removido o processo acontece da mesma forma: ordem reversa;
- A utilização de réplicas também é utilizada para garantir a disponibilidade;

Amazon Dynamo DB

Disponibilidade:

- Réplicas:
 - Pode ser configurado para perceber que os nós estão em mais de um *data center*;
 - Inconsistências entre as réplicas são automaticamente gerenciadas:
 - Reconciliação sintática: não existe conflitos e a última versão pode ser considerada
 - Reconciliação semântica: há conflitos e será resolvido pelo cliente;

Amazon Dynamo DB

Consistência:

- A resposta de uma solicitação de escrita possui durabilidade (escrita realizado em múltiplos servidores);
- Existe um período de tempo necessário para a propagação da atualização em todas as réplicas: consistência eventual;
- Há a chance de algumas atualizações não serem realizadas em algumas réplicas, em caso de falhas de nós;

Cláudio Lúcio

35

Amazon Dynamo DB

Consistência:

- Oferece a possibilidade de uma leitura requisitar o dado mais atualizado possível: Leitura consistente/resolução de conflitos;
- Para o caso de detecção e resolução de conflitos utiliza o vetor de relógios;
- Toda operação de *update* necessita de um contexto: o vetor de relógios, para determinar qual réplica será atualizada;

Amazon Dynamo DB

Consistência:

- Atualização condicional
 - Especificar uma condição na atualização de um item: item só é escrito se a condição é atendida;
 - Exemplo (*lost-update*):
 - Na atualização o cliente verifica se a versão anterior (cópia obtida) continua 'vigente' no servidor;
 - Se sim faz a atualização, caso contrário recebe a nova versão e repete a operação de atualização;
- Possui a funcionalidade de "*atomic counter*": adicionar ou subtrair de contador de forma atômica;

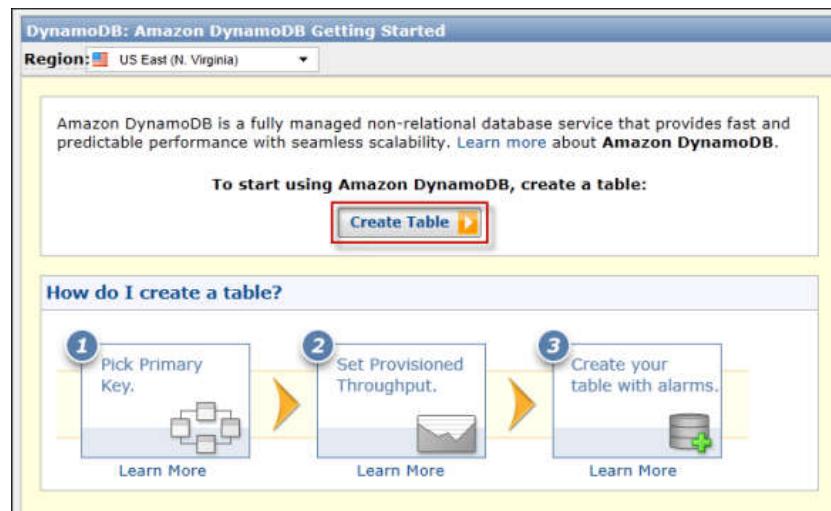
Cláudio Lúcio

37

Amazon Dynamo DB

Console de administração - exemplo:

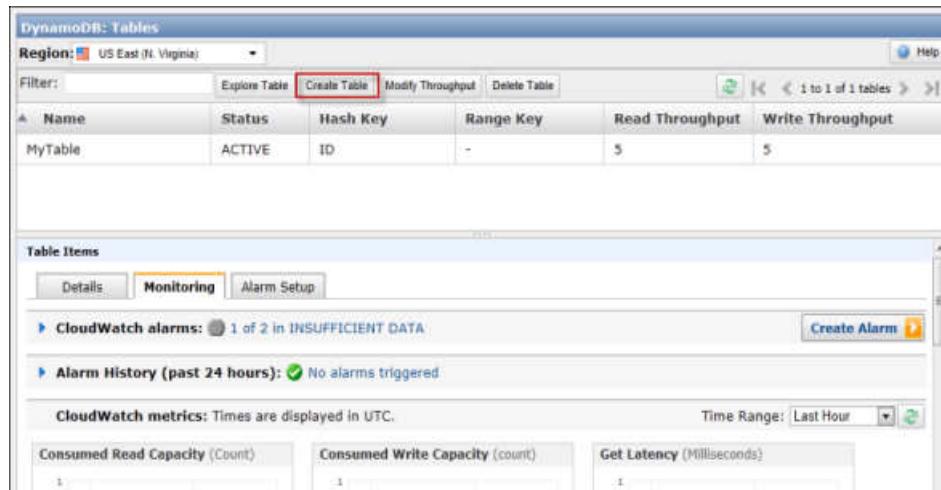
- Criação de tabelas:



Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:



DynamoDB: Tables

Region: US East (N. Virginia)

Filter: Explore Table Create Table Modify Throughput Delete Table

Name	Status	Hash Key	Range Key	Read Throughput	Write Throughput
MyTable	ACTIVE	ID	-	5	5

Table Items

Details Monitoring Alarm Setup

CloudWatch alarms: 1 of 2 in INSUFFICIENT DATA Create Alarm

Alarm History (past 24 hours): No alarms triggered

CloudWatch metrics: Times are displayed in UTC. Time Range: Last Hour

Consumed Read Capacity (Count): 1 Consumed Write Capacity (Count): 1 Get Latency (Milliseconds): 1

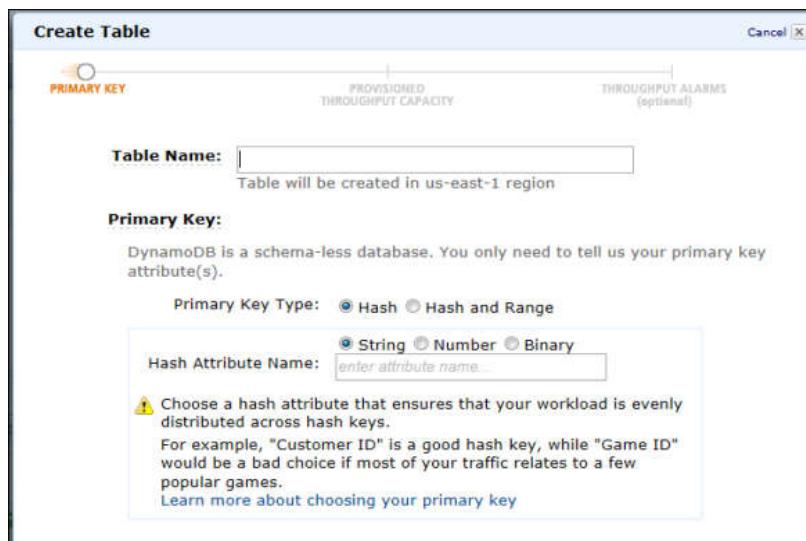
Cláudio Lúcio

39

Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:



Create Table

PRIMARY KEY PROVISIONED THROUGHPUT CAPACITY THROUGHPUT ALARMS (optional)

Table Name: Customer

Table will be created in us-east-1 region

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: Hash Hash and Range

Hash Attribute Name: Customer ID

⚠ Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.

For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.

Learn more about choosing your primary key

Cláudio Lúcio

Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: Hash Hash and Range

String Number Binary

Hash Attribute Name:

⚠ Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.

For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.

[Learn more about choosing your primary key](#)

Cláudio Lúcio

41

Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:

Primary Key:

DynamoDB is a schema-less database. You only need to tell us your primary key attribute(s).

Primary Key Type: Hash Hash and Range

String Number Binary

Hash Attribute Name:

String Number Binary

Range Attribute Name:

⚠ Choose a hash attribute that ensures that your workload is evenly distributed across hash keys.

For example, "Customer ID" is a good hash key, while "Game ID" would be a bad choice if most of your traffic relates to a few popular games.

Cláudio Lúcio

Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:

The screenshot shows the 'Create Table' wizard with the 'PROVISIONED THROUGHPUT' tab selected. It includes fields for 'Read Capacity Units' and 'Write Capacity Units', both with placeholder text 'enter throughput...'. A note at the top says 'Help me calculate how much throughput capacity I need to provision'.

Cláudio Lúcio

43

Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:

The screenshot shows the 'Create Table' wizard with the 'THROUGHPUT ALARMS (optional)' tab selected. It includes a checkbox 'Use Basic Alarms' which is checked. Below it, a dropdown menu says 'Notify me when my table's request rates exceed 80% of Provisioned Throughput for 60 minutes.' A section 'Notification will be sent when:' lists three conditions: 'Read Capacity Units consumed > 8', 'or', and 'Write Capacity Units consumed > 4'. A field 'Send notification to:' contains the email 'awsAccount@domain.com'.

Cláudio Lúcio

Amazon Dynamo DB

Console de administração - exemplo:

- Criação de tabelas:

Name	Status	Hash Key	Range Key	Read Throughput	Write Throughput
Forum	ACTIVE	Name	-	10	5
ProductCatalog	ACTIVE	Id	-	10	5
Reply	ACTIVE	Id	ReplyDateTime	10	5

Cláudio Lúcio

45

Amazon Dynamo DB

Console de administração - exemplo:

- Consulta no console:

Name	Status	Hash Key	Range Key	Read
Reply	ACTIVE	Id	ReplyDateTime	10

Vídeo de demonstração da Amazon



Atividade

Reflexão

- Simplicidade da abordagem chave valor;
- O termo ausência de esquema faz sentido para os bancos de dados baseados em chave-valor?
- As funções **Map** e **reduce** são aplicáveis na abordagem chave valor?
- Amazon DynamoDB: disponibilidade, tolerância a partição e consistência.
- Consegue pensar em uma utilização com o Amazon DynamoDB?



Bancos de dados baseados em documentos

Cláudio Lúcio

1

Bancos de dados de Documentos

Agenda

- Conceitos
- CouchDB



Bancos de dados baseados em documentos

Conceitos

Cláudio Lúcio

3

Conceitos

- Não são sistemas de gerenciamento de documentos (CMS);
- Alguns consideram os banco de dados baseados em documentos como a próxima evolução do bancos chave valor;
- O termo documento = conjunto estruturado de chaves e valores;



Conceitos

- Bancos de dados relacionais possuem forte esquema(tabelas e colunas)
- Bancos de dados chave valor trabalham com valores tratados na forma de difícil interpretação;
- Bancos de dados de documentos optam por um terceiro caminho: dados estão contidos em documentos que não possuem um esquema fixo;
- Este esquema é conhecido pelas aplicações que consomem os dados, assim como pelo próprio banco de dados;

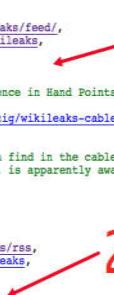
Cláudio Lúcio

5

Conceitos

- O documento é tratado como um todo e não como chave valor;
- Oriundo do JSON (JavaScript Object Notation);

```
- {  
    feedTitle: "Wikileaks on The Huffington Post",  
    feedUrl: http://www.huffingtonpost.com/tag/wikileaks/feed/,  
    websiteUrl: http://www.huffingtonpost.com/tag/wikileaks,  
    whenLastUpdate: "Fri, 17 Dec 2010 22:32:06 GMT",  
    - item: {  
        id: "0024801",  
        title: "David Danzig: Wikileaks Cables: Evidence in Hand Points to &quot;Masterminds&quot; Behind Assassination of Prominent Indonesian Rights Activist",  
        link: http://www.huffingtonpost.com/david-danzig/wikileaks-cables-evidence_b_798487.html,  
        permalink: "",  
        pubDate: "Wed, 01 Dec 1999 00:00:00 GMT",  
        body: "Among the handful of bombshells one can find in the cables that went back and forth between the U.S. State Department and embassy staff in Jakarta is this: The U.S. is apparently aware of evidence linking a high level Indonesian security official to the assassination of Munir..."  
    },  
    - {  
        feedTitle: "Media: WikiLeaks | guardian.co.uk",  
        feedUrl: http://www.guardian.co.uk/media/wikileaks/rss,  
        websiteUrl: http://www.guardian.co.uk/media/wikileaks,  
        whenLastUpdate: "Fri, 17 Dec 2010 22:02:06 GMT",  
        - item: [  
            - {  
                id: "0024745",  
                title: "Q&A: Julian Assange allegations",  
                link: http://www.guardian.co.uk/media/2010/dec/17/julian-assange-q-and-a,  
                permalink: http://www.guardian.co.uk/media/2010/dec/17/julian-assange-q-and-a,  
                pubDate: "Fri, 17 Dec 2010 21:50:52 GMT",  
                body: "Despite three legal hearings, there remains a lack of consensus about the status of charges against the WikiLeaks founder. Has Assange been charged with an offence, or are these just accusations? Assange's lawyers insist he has not been charged and that the warrant against him is...",  
                - enclosure: {  
                    url: http://static.guim.co.uk/sys-images/Guardian/Pix/pictures/2010/12/17/1292619509205/Julian-Assange-003.jpg,  
                    type: "Image/jpeg",  
                    length: 6527  
                },  
            },  
            - {  
                id: "0024714",  
                title: "Wikileaks cables: Sudanese president 'stashed $9bn in UK banks'",  
                link: http://www.guardian.co.uk/world/2010/dec/17/wikileaks-sudanese-president-cash-london,  
                permalink: http://www.guardian.co.uk/world/2010/dec/17/wikileaks-sudanese-president-cash-london,  
                pubDate: "Fri, 17 Dec 2010 21:03:24 GMT",  
                body: "Speculation that Omar al-Bashir siphoned $9bn in oil money and deposited it in foreign accounts could fuel calls for his arrest. Omar al-Bashir, the Sudanese president, has siphoned as much as $9bn out of his impoverished country, and much of it may be stashed in London banks..."  
            }  
        ]  
    }  
}
```



Cláudio Lúcio

Conceitos

- Utilizam o termo de coleção de documentos;
- No nível da coleção é possível agrupar vários documentos;
- Diferentemente dos bancos de dados chave valor, no bancos de dados baseados em documentos é possível indexar os documentos com base nas suas propriedades;

Cláudio Lúcio

7

Conceitos

- MongoDB:
 - Site: <http://www.mongodb.org>
 - Histórico: Criado pela empresa 10gen ;
 - Linguagem: foi implementado em C++;
 - Métodos de acesso: possui uma linha de comando JavaScript e possui bibliotecas de acesso em C, C#, C++, Erlang. Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, e Scala.
 - Vantagem: SQL-*like*.

Cláudio Lúcio

Conceitos

- CouchDB:
 - Site: <http://couchdb.apache.org>
 - Histórico: criado em 2005 e incubado no projeto Apache em 2008 ;
 - Linguagem: foi implementado em Erlang com algumas implementações em C e JavaScript;
 - Métodos de acesso: REST(*Representational state transfer*);
 - Utilização: Apple, BBC, Facebook (aplicativos) e outros;

Cláudio Lúcio

9



CouchDB

- Características:

- CouchDB = “Cluster of unreliable commodity hardware” Database;
- Descendente do 'Lotus notes';
- Utiliza a filosofia RESTful:
 - Arquitetura de aplicativos da Web fracamente acoplados que contam com recursos nomeados: Localizador Uniforme de Recursos (URL), Identificador Uniforme de Recursos (URI) e Nome de Recurso Uniforme (URN);
 - RESTful possibilita sistemas altamente escaláveis, enquanto são fracamente acoplados aos dados subjacentes;

Cláudio Lúcio

11

CouchDB

- Características:

- Utiliza a filosofia RESTful:
 - Utilizam aspectos do protocolo HTTP como pedidos GET e POST;
 - Pedidos são mapeados para necessidades de aplicativo de negócios , como CRUD: *create, read, update e delete*;

Tarefa do aplicativo	Comando HTTP
Create	POST
Read	GET
Update	PUT
Delete	DELETE

CouchDB

- Características:

- Em resumo: banco de dados de documentos que é acessado via RESTful HTTP;
- Documentos são *schema-free*;
- Considerado por alguns como banco de dados semi-estruturado;
- Com a base de documentos:
 - Funções de seleção e agregação dos documentos podem ser criadas no paradigma '*map reduce*' → visões do banco de dados;
 - São replicadas entre '*data nodes*';

Cláudio Lúcio

13

CouchDB

- Características:

- Cada documento possui um único ID;
- Utiliza o conceito de múltiplas versões do mesmo documentos;
- Para as várias versões de um mesmo documento o banco de dados detecta conflitos e delega para os clientes a resolução;
- Possui um mecanismo simples de segurança de documentos e usuários;
- Funções especiais *JavaScript* podem ser escritas para serem executadas antes da gravação dos arquivos no disco. Se houver erro retorna e não grava o arquivo;

CouchDB

- Modelos de dados
 - Documentos:
 - Conjunto de nomes e valores
 - Os nomes devem ser únicos em um documento;
 - Os valores podem ser dos tipos: carácter , número, lógico, data, lista ordenada e outros;
 - Pode haver links entre os documentos (URL ou URI), mas não há validação da referência;
 - O ID do documento é uma chave com 128 bits e o número da revisão contém 32 bits;

Cláudio Lúcio

15

CouchDB

- Modelos de dados
 - Visões
 - É a forma de apresentar 'consultas' a partir dos dados semi-estruturados;
 - Ex.: obter o número de documentos por tipo do documentos. Para o banco de dados eles continuam sendo apenas documentos;
 - As visões são definidas em JavaScript;
 - As funções são armazenadas no bancos de dados e são funções *map/reduce*;
 - A função *map*: recebe o documento como parâmetro e realiza operações nos dados, logo após emite os dados: document id, chave e valor;
 - A função *reduce*: é opcional e pode fazer uma agregação nos dados;

Cláudio Lúcio

CouchDB

- Modelos de dados
 - Visões
 - São executadas e materializadas quando as visões são acionadas pela primeira vez;
 - Existe um processo chamado 'view-builder': responsável por atualizar as *views*:
 - Compara o id do documento e revisão das views com o banco de dados;
 - Verifica documentos excluídos e criados desde a atualização da visão;
 - Atualização da visão é eficiente: 'append-only';
 - Durante a atualização da visão, a versão anterior continua disponível;
 - Se um documento possui uma nova revisão, somente a função *map* para aquele documento é executada;

Cláudio Lúcio

17

CouchDB

- Modelos de dados
 - Visões
 - As visões são armazenadas em uma estrutura B-tree, da mesma forma que os documentos;
 - As visões são armazenadas em arquivos separados e ajustados para maior desempenho;
 - Não existe limitação para o número de visões por bancos de dados;

CouchDB

- Modelos de dados

- Bancos de dados

- Cada banco de dados consiste em um conjunto de documentos – o servidor gerencia os ID's dos documentos;
 - Um '*server node*' pode conter mais do que banco dados;
 - Os documentos são retidos no disco como arquivos XML ou serializados no formato JSON;
 - Uma indexação B-TREE é feita para os ID's dos documentos e número de revisão;
 - Realiza periodicamente (agendada ou de acordo com *threshold*) a compactação do banco de dados (espaço em disco) – *Append-only*; É realizado sem parar o banco;

CouchDB

- Escalabilidade (Tolerância a Partição):

- Não possui papéis especiais para os nós: '*server nodes*';
 - Faz replicação de documentos ou partes de documentos entre diferentes '*server nodes*';
 - O *cluster* possui um processo de replicação que opera de forma incremental (avaliando as versões) e por documento (parcial);
 - A distribuição de documentos por nós pode ser manual ou automática;
 - Utiliza o algoritmo '*consistent hashing*' para atribuir os dados para os nós ;

CouchDB

- Escalabilidade (Tolerância a Partição):
 - Os documentos são armazenados nos nós em particões: metadados de documento ID por partição;
 - No caso de novas máquinas as partições são movidas ou replicadas: de acordo com o caso usando '*consistent hashing*' ;

Cláudio Lúcio

21

CouchDB

- Disponibilidade:
 - Para a tolerância a falhas recomenda-se a utilização das replicas de documentos e múltiplos bancos de dados;
 - Neste configuração o *cluster* possui:
 - Elevada disponibilidade;
 - Faz balanceamento de carga;
 - A adição de '*servers nodes*' melhora o desempenho e a tolerância a falhas;

CouchDB

- Consistência:
 - Usa o MVCC – *Multi-Version Concurrency Control*: todas as versões dos documentos nos nós;
 - MVCC = '*append only*';
 - A consistência, em um cenário *cluster*, é eventual;
 - A resolução de conflitos deve ser feita em tempo de leitura:
 - Regras pré-definidas na replicação;
 - Pelo próprio cliente no momento da leitura;
 - Observe que na resolução de conflitos todas as versões dos dados estarão disponíveis;

Cláudio Lúcio

23

CouchDB

- Administração de usuários
 - Modelo de controle de acesso de documentos: listas de usuários autorizados a ler o documento ('*readers*');
 - Esta mesma restrição é válidas para as visões que acessam o documento;
 - Existe o perfil do administrador: criar bancos e usuários;

Atividade e Reflexão

Atividade

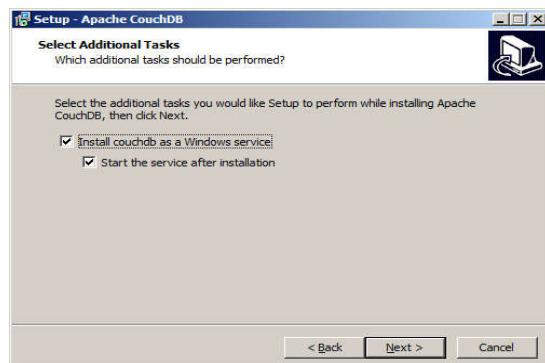
- Demonstração utilizando o CouchDB
- Reflexão
 - Faz sentido uma evolução do chave valor para os documentos?
 - O termo 'schemaless' é válido para os bancos de dados de documentos ou eles são semi-estruturados?
 - A importância do paradigma Map/Reduce para o CouchDB e bancos de dados de documentos?
 - Importância de ser avaliar o teorema CAP em uma solução NoSQL?



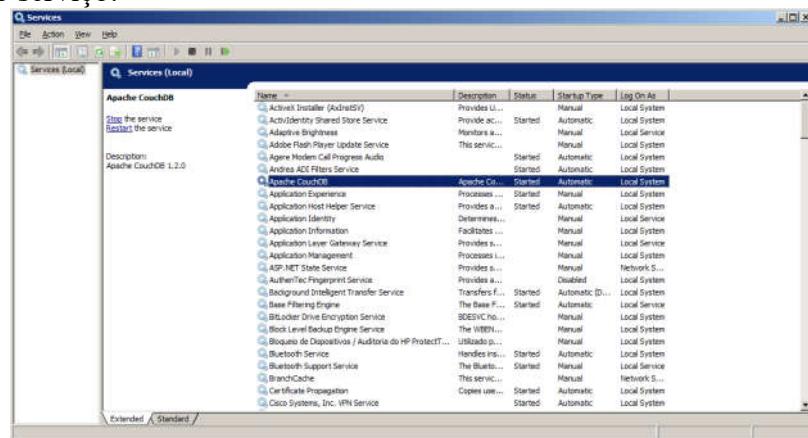
1) Instalação do CouchDB:

Download: http://ftp.unicamp.br/pub/apache/couchdb/packages/win32/1.2.0/setup-couchdb-1.2.0_otp_R14B04.exe

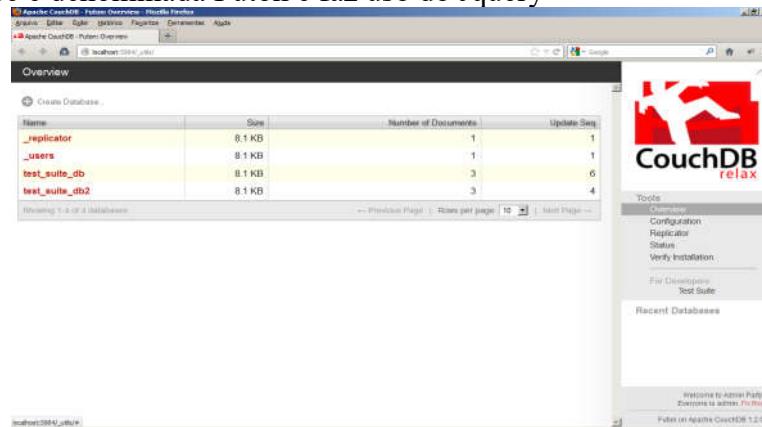
Instale o produto:



2) Verifique o serviço:



3) Acesse o endereço: http://localhost:5984/_utils/ - Utilize o Mozilla Firefox:
Esta interface é denominada Futon e faz uso do Jquery



4) Usuários

- Por padrão o banco vem totalmente aberto;
- Clique na opção “fix it”;

The screenshot shows the Apache CouchDB Futon Overview interface. On the left, there's a table listing four databases: '_replicator', '_users', 'test_suite_db', and 'test_suite_db2'. The '_replicator' database has 0.1 KB size, 1 document, and update seq 1. The '_users' database has 8.1 KB size, 1 document, and update seq 1. The 'test_suite_db' database has 8.1 KB size, 3 documents, and update seq 6. The 'test_suite_db2' database has 8.1 KB size, 3 documents, and update seq 4. On the right side, there's a sidebar with a CouchDB logo and links for 'Tools', 'Configuration', 'Replicator', 'Status', 'Verify Installation', 'For Developers', and 'Test Suite'. Below the sidebar, a 'Recent Databases' section lists the same four databases. At the bottom right of the main area, a status bar displays the message 'Welcome to Admin Panel'.

- Digite o nome do usuário administrador e a senha

The screenshot shows the Apache CouchDB Futon Overview interface with a 'Create Server Admin' dialog box open. The dialog box contains instructions about server admins and their roles. It asks for a 'Username' (which is left empty) and a 'Password'. Below the password field, there's a note about non-admin users having read and write access to all databases. At the bottom of the dialog box are 'Create' and 'Cancel' buttons. The background shows the list of databases from the previous screenshot.

5) Crie um banco de dados chamado 'banconosql' Clique na opção 'Create Database';

The screenshot shows the Apache CouchDB Futon Overview interface with a 'Create New Database' dialog box open. The dialog box has a note about database naming rules (only lowercase characters, underscores, and hyphens are allowed). It asks for a 'Database Name' (which is set to 'banconosql'). At the bottom are 'Create' and 'Cancel' buttons. The background shows the list of databases from the previous screenshots.

6) Crie um documento chamado produto

- Clique em “New Document”;
- Clique em “Add Field” para adicionar dados para o documento JSON.
- Adicione a chave “nome” com o valor “Curso de banco de dados NoSQL”;
- Clique em “Save Document”;
- Veja que o campo `_rev` é automaticamente gerado;

- Faça uma alteração no campo nome para 'Curso de banco de dados NoSQL CouchDB';
- Crie mais um campo: 'carga_horaria' e preencha com o valor 10
- Clique em “Save Document”;
- Veja o novo número da revisão;
- Ainda é possível verificar as versões anteriores e atual: “Previous Version” e “Next Version”

7) Navegue para o banco de dados e veja o documento criado:

8) Crie mais outros três documentos:

nome = 'Curso de banco de dados – DynamoDB';
carga_horaria= 3

nome = 'Curso de banco de dados – Voldemort';
carga_horaria= 6

nome = 'Curso de banco de dados – Cassandra';
carga_horaria= 1
autor = escreva seu nome

9) Criação da visão para saber o total da carga horária dos cursos

- Clique na opção 'Views' e selecione o item temporary view;
- No item 'map function' altere para:

```
function(doc) {
    emit(doc.nome, doc);
}
```

- Clique em 'run' para visualizar o resultado;

- Novamente, no item 'map function' altere para:

```
function(doc)
{
    if (doc.autor === "claudio"){
        emit(doc.nome, doc);
    }
}
```

- Clique em 'run' para visualizar o resultado;

- Novamente, no item 'map function' altere para:

```
function(doc) {
    emit(doc.nome, doc.carga_horaria);
}
```

- Altere a função Reduce para :

```
function (keys, carga_horaria) {
    return sum(carga_horaria);
}
```

- Clique em 'run' para visualizar o resultado;

- Atente para a opção “Grouping”

- Clique na opção 'Save as' e digite
Design Document: 'Total';
View Name: Total Carga Horaria
- Clique na opção 'Design documents' e visualize a view recém-criada;

10) Clique na opção 'Configuration' – para visualizar algumas opções de configuração

Section	Option	Value
admins	admin	-hashed-62e508fc865ff12998af2eb5d547e0ff397b6102, b547d1298ef19d5390611713f6d105c0
attachments	compressible_types	text/*, application/javascript, application/x-javascript, application/x-xml
compaction_daemon	compression_level	8
couch_httpd_auth	check_interval	300
couch_httpd_auth	min_file_size	131072
couch_httpd_auth	allow_persistent_cookies	false
couch_httpd_auth	auth_cache_size	50
couch_httpd_auth	authentication_db	_users
couch_httpd_auth	authentication_redirect	/_utils/session.html
couch_httpd_auth	require_valid_user	false
couch_httpd_auth	secret	847104ff73cb1fa06004a490ee07f9c6
couch_httpd_auth	timeout	600
couch_httpd_oauth	use_users_dbi	false
couchdb	database_dir	./var/lib/couchdb
couchdb	delayed_commits	true
couchdb	file_compression	snappy
couchdb	max_dbs_open	100
couchdb	max_document_size	4294967295
couchdb	os_process_timeout	5000
couchdb	uri_file	./var/run/couchdb/couch.uri
couchdb	util_driver_dir	./lib/couch-1.2.0/priv/lib

11) Clique na opção 'Replicator' – Replicação dos banco de dados

- Crie um novo banco de dados vazio: 'replica'
- Clique me 'Replicator' e inicie o processo de replicação. Marque a opção 'continous':

The screenshot shows the 'Replicator' configuration screen. It has two dropdown menus: 'Replicate changes from:' (set to 'Local Database: banconosql') and 'to:' (set to 'Local database: replica'). Below these are checkboxes for 'Continuous' (unchecked) and 'Replicate' (checked). A status bar at the bottom shows the event log: "ok=true, _local_id='206ccc52600770242c301cd8e645a20+continuous'".

- Clique e veja o conteúdo do banco 'replica'
- Crie um novo documento no banco 'banconosql'
 nome = 'Teste de replica';
 carga_horaria= 1
- Verifique novamente o banco 'replica' para visualizar a execução da replicação;

NoSQL MongoDB MapReduce Documentos

Cassandra BerkeleyDB Paralelismo chave-Valor

Python Processamento DFS Colunar

Bancos de dados baseados em Colunas

Cláudio Lúcio

1

Bancos de dados de Colunares

Agenda

- Conceitos
- Cassandra



Bancos de dados baseados em Colunas

Conceitos

Cláudio Lúcio

3

Conceitos

- Origem: *Analytics* e *Business Intelligence*;
- Exemplo de produto: Sybase IQ - 2002
- A proposta é um armazenamento em colunas:

Country	Product	Sales
US	Alpha	3.000
US	Beta	1.250
JP	Alpha	700
UK	Alpha	450

Row 1	US	Alpha	3.000
-----	US		
Row 2	Beta		1.250
-----	JP		
Row 3	Alpha		700
-----	UK		
Row 4	Alpha		450

Country	US	US	JP	UK
Product	Alpha	Beta	Alpha	Alpha
Sales	3.000	1.250	700	450

- Orientada a linhas: fácil de adicionar novas linhas, leitura de colunas (dados) adicionais;
- Orientado a colunas: Somente lê colunas necessárias, necessidade de 'busca' na adição de novas linhas;

Conceitos

- Abordagem economiza espaço. Se a coluna for nula, não será armazenada.
- Orientado para leituras (*Data Warehouse*):
 - Desempenho para várias agregações e leituras;
 - Escritas são executadas em lote;
 - Normalmente está no Top 3 do TPC-H (Exasol, ParAccel e Kickfire);

Cláudio Lúcio

5

Conceitos

- Cada unidade do dado (linha no SGBDR) é um conjunto de pares chave valor ;
- A unidade do dado é identificada com a ajuda de uma chave primária ('row key');
- Um conjunto de colunas é chamado de família de colunas;
- Dentro da família de colunas, as chaves primárias são armazenadas de forma ordenada;

Cláudio Lúcio

Conceitos

- Considere o seguinte exemplo

- Nome: Fulano
 - Sobrenome: de tal
 - Tratamento: Dr.
 - Endereço: Rua novembro
 - Número: 34
 - CEP: 30000-000
- Nome:Ciclano
 - Sobrenome: di tal
 - Endereço: Av. Sem numero
 - Número: 56
- Nome: Diclano
 - Sobrenome: sem tal

Cláudio Lúcio

7

Conceitos

Família de colunas

↓ Ordenado pela chave

1	Nome	Sobrenome	Tratamento
	Fulano	de tal	Dr.
2	Nome	Sobrenome	
	Ciclano	di tal	
3	Nome	Sobrenome	
	Diclano	sem tal	

Família de colunas

↓ Ordenado pela chave

1	Endereço	Número	CEP
	Rua Novembro	34	30000-000
2	Endereço	Número	
	Av. Sem numero	56	

Conceitos

- Colunas
 - Cada coluna e linha possui controle de versão;
 - Milhões de colunas;
 - As colunas podem ficar em memória ou podem ser compactadas;
- Família de colunas
 - Na prática as famílias de colunas não ficam fisicamente isoladas;

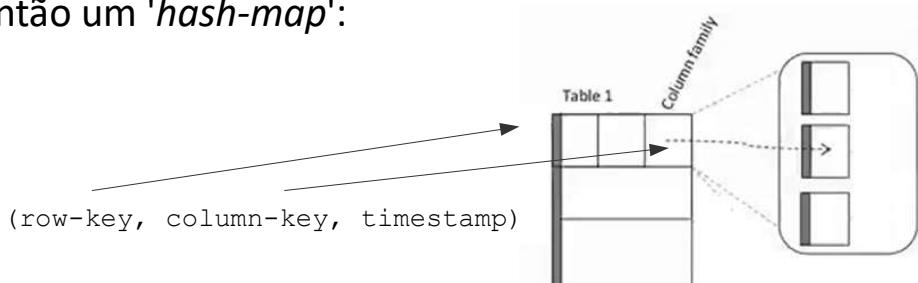
Cláudio Lúcio

9

Conceitos

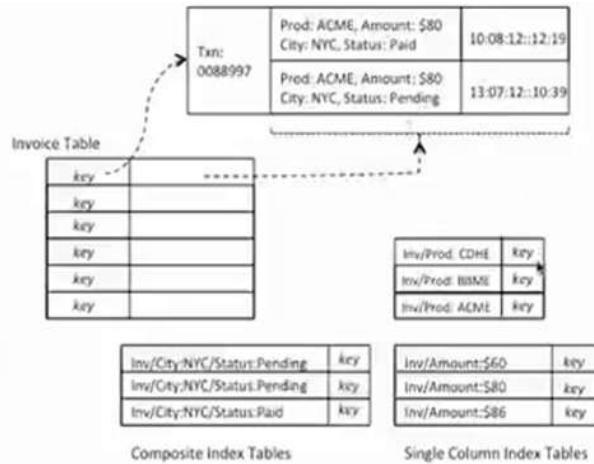
Família de colunas

- Definida no momento de criação dos dados;
- É recomendável que não tenha alterações muito frequentes;
- Cada família de colunas é uma estrutura chave valor. Cada família de colunas possui uma chave e a linha é então um '*hash-map*':



Conceitos

- As chaves ordenadas propiciam a eficiência para leitura;
- No acesso aos dados (leitura) além da chave ordenada, as colunas podem conter estruturas para melhorar as consultas:



Cláudio Lúcio

11

Conceitos

- Lógica de acesso aos dados
 - $(Table, RowKey, Family, Column, Timestamp) \rightarrow Value$
 - $\text{SortedMap} < \text{RowKey}, \text{List} < \text{SortedMap} < \text{Column}, \text{List} < \text{Value}, \text{Timestamp} >>>$
- O primeiro '*SortedMap*' é a tabela contendo a lista de família de colunas;
- A família de colunas por sua vez, contém outro '*SortedMap*' com as colunas e seus valores e versões;

Conceitos

- Armazenamento:

- Todos os dados pertencentes a uma chave e são retidas em conjunto;
- O armazenamento dos dados é ordenado, mas o processo de partição entre os nós pode ser aleatório ou não;
- Normalmente faz uso de DFS, desta forma a tolerância a falhas pode ser garantida com a replicação de dados em nós diferentes do '*cluster*';
- Arquivos de dados e logs são armazenados nos 'data nodes';

Cláudio Lúcio

13

Conceitos

- Arquitetura:

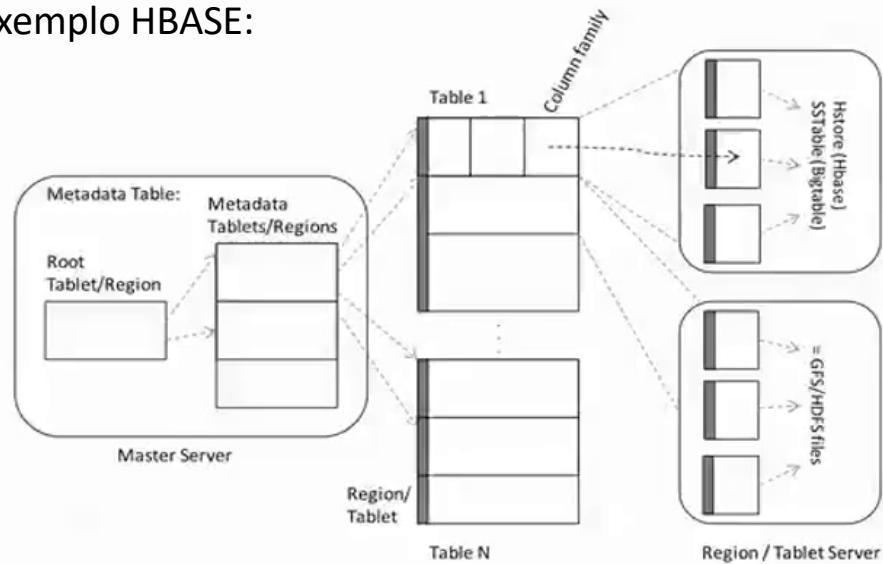
- Alguns produtos utilizam os '*server nodes*' e em outros não há figura de um controlador central;
- Retêm metadados para:
 - Local de armazenamento de cada tabela;
 - Servidores disponíveis;
 - Informações de esquema: famílias de colunas;
 - Listas de controles de acesso;
- No Google *Bigtable* um serviço central consiste de um *cluster* com 5 réplicas ativas. Em uma nova requisição, uma delas é eleita como '*master*' e faz toda execução;

Cláudio Lúcio

Conceitos

- Arquitetura:

Exemplo HBASE:



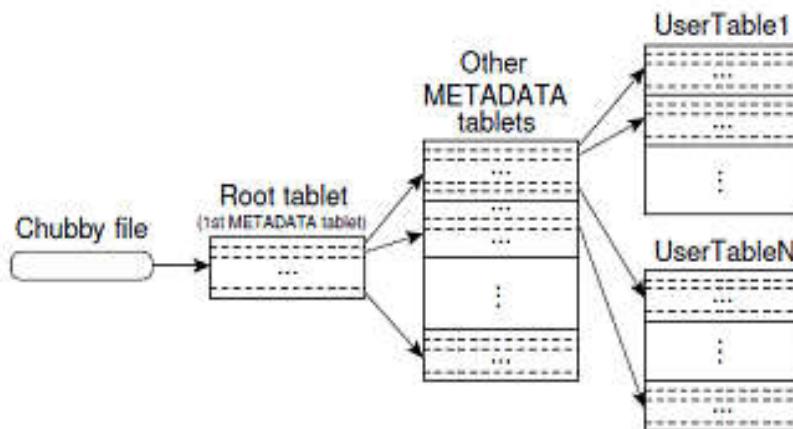
Cláudio Lúcio

15

Conceitos

- Arquitetura:

Exemplo Google BigTable:



Conceitos

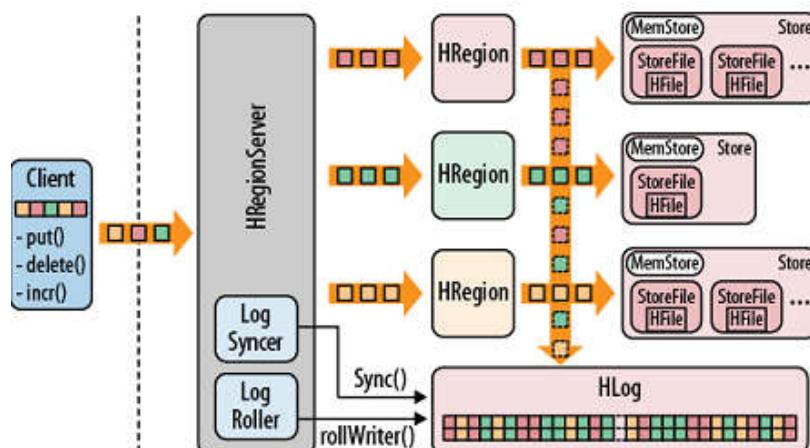
- Arquitetura:
 - Possui arquivos de log (comum em vários produtos);
 - Também é comum nos bancos colunares o uso de '*write ahead log*';
 - Exemplo HBASE:
 - Cliente modifica o dado(*put()*, *delete()*, ...);
 - Modificações são tratadas para estrutura chave valor;
 - Dados são encontrados no Servidor(HRegionServer) e no arquivo(HRegion);
 - Objetos são escritos no Log e na memória(MemStore);

Cláudio Lúcio

17

Conceitos

- Arquitetura:
 - Exemplo HBASE:



Conceitos

- HBASE:
 - Site: <http://hbase.apache.org>
 - Histórico: Criado pela empresa Powerset. Doado a fundação Apache antes de ser adquirida pela Microsoft em 2007;
 - Linguagem: foi implementado em Java;
 - Métodos de acesso: REST(*Representational state transfer*) e JAVA API; Outro projeto Apache introduz SQL neste banco de dados:Hive (<http://hive.apache.org>)
 - Utilização: Facebook, StumbleUpon, Mahalo, Yahoo e outros;

Cláudio Lúcio

19

Conceitos

- Cassandra:
 - Site: <http://cassandra.apache.org/>
 - Histórico: Desenvolvido no Facebook. Projeto teve seu código aberto em 2008. Foi doado para a fundação Apache;
 - Linguagem: foi implementado em Java;
 - Métodos de acesso: JAVA API e outras linguagens: Python, Grails, PHP, .NET e Ruby; Possui o CQL – 'SQL-like' para manipulação de dados
 - Utilização: Facebook, Twitter;



Bancos de dados baseados em Colunas

Cassandra

Cláudio Lúcio

21

Cassandra

- Características:
 - Utiliza conceitos tanto do *Amazon DynamoDB* quanto do *Google Bigtable*;
 - Em alguns livros e artigos é tido como um banco de dados chave valor;
 - Possui uma API para os clientes simplificada (*get*, *insert* e *delete*);
 - Não possui nó central, utiliza um protocolo *Gossip*, em que troca de mensagens entre os servidores de forma intensiva;
 - Todos os servidores possuem log;

Cassandra

- Características:
 - Possui também uma estrutura de dados em memória (*Memtable*). Dados são escritos no disco quando a memória esta cheia;
 - Nós podem ser adicionados em tempo real ao *cluster*;
 - Não utiliza uma DFS como o Google Bigtable. Dados são gravados localmente e, se for o caso, replicados;
 - Os arquivos de dados são compactados com certa periodicidade, por um processo de plano de fundo;
 - Possui uma linguagem '*SQL like*': Cassandra Query Language -CQL (DDL, DML);

Cláudio Lúcio

23

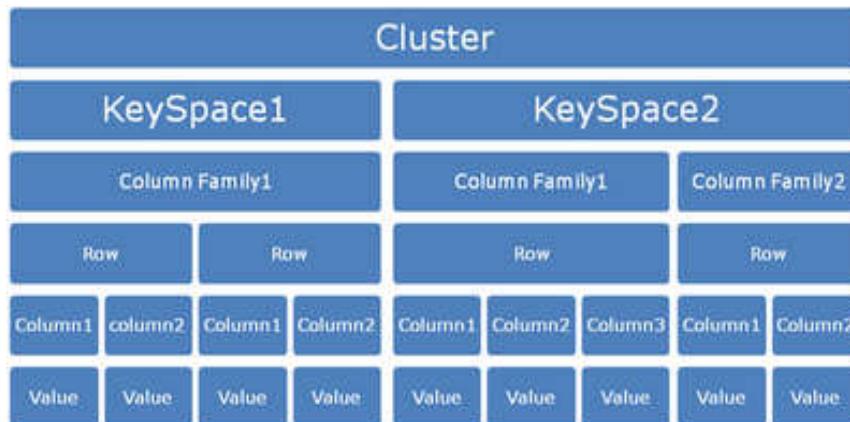
Cassandra

- Características:
 - Um nó servidor no *cluster* cassandra executa alguns módulos:
 - Particionamento;
 - Mecanismo de armazenamento;
 - Detecção a falha e nó do cluster;
 - Todos os módulos são implementados em Java;
 - No caso de uma nova requisição;
 - Identifica-se qual nó possui a chave;
 - A requisição é enviada para o nó identificado e a requisição é atendida;
 - Caso haja falha na resposta, indica para o cliente e procura pela última réplica e responde;

Cláudio Lúcio

Cassandra

- Modelos de dados:



Cláudio Lúcio

25

Cassandra

- Modelos de dados:

- Coluna

- É o elemento mais granular na hierarquia do Cassandra;
 - Possui a seguinte estrutura:

```
struct Column {  
    1: binary  
    2: binary  
    3: i64  
}
```

- O valor '*timestamp*' é fornecido pelo cliente e pode ser utilizado para resolução de conflitos;
 - O nome e valor são binários, podem ser serializados utilizando strings UTF8;

Cassandra

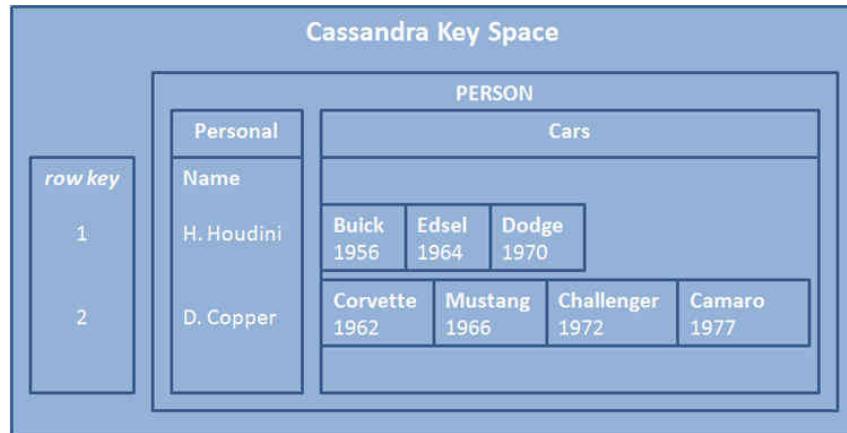
- Modelos de dados:
 - Famílias de colunas
 - É um repositório para linhas, analogamente, seriam as tabelas nos SGBDR's;
 - Cada linha em um família de colunas é referenciada por sua chave;
 - O armazenamento das chaves nas famílias de colunas é ordenado, sendo configurável a ordenação aplicada (ASCII, UTF-8, Long, etc.);
 - Colunas relacionadas (serão acessadas em conjunto) devem ser armazenadas na mesma família de colunas;

Cláudio Lúcio

27

Cassandra

- Modelos de dados:
 - Famílias de colunas – exemplo dinâmico



Cláudio Lúcio

Cassandra

- Modelos de dados:
 - Linhas
 - Cada família de colunas é armazenada em um arquivo separado e o arquivo é ordenado por linhas;
 - A chave da linha é o que determina em qual nó do *cluster* será armazenado o dado;
 - Para cada chave pode haver dados em múltiplas famílias de colunas;

Cláudio Lúcio

29

Cassandra

- Modelos de dados:
 - KeySpace
 - Pode ser considerado como esquema ou banco de dados (primeiro *hash map* do armazenamento no Cassandra);
 - Armazena as famílias de colunas;

Cassandra

- Escalabilidade (tolerância a partição):
 - Não possui papéis especiais para os nós: '*server nodes*';
 - Os dados de uma família de colunas são particionados e distribuídos entre os nós utilizando o algoritmo '*consistent hashing*' que preserva a ordem das chaves;
 - O algoritmo '*consistent hashing*' possui uma alteração comparada com a do Amazon DynamoDB: não utiliza o conceito de '*virtual node*';
 - Faz análise constante das cargas nos servidores e se necessária altera a ordem dos servidores no anel para o '*consistent hashing*

Cláudio Lúcio

31

Cassandra

- Escalabilidade (tolerância a partição):
 - Os particionamentos são feitos pelas chaves: aleatório ou ordenados;
 - O particionamento dos dados é controlado por um arquivo de configuração : *cassandra.yaml*;
 - Uma vez que um *cluster* é iniciado com uma opção de particionamento, o mesmo não pode ser alterado sem que haja a recarga dos dados no *cluster*;

Cassandra

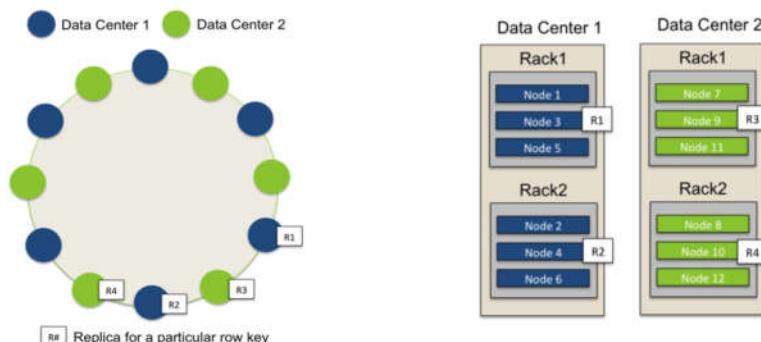
- Disponibilidade:
 - Utiliza o protocolo *Gossip* entre os nós do cluster;
 - No *cluster* cada nó tenta localizar e detectar se outro nó está ativo/desativo (evitar tentativas de requisições que vão falhar);
 - As réplicas são controladas por um fator que é chamado de '*replication factor*';
 - O fator de replicação é determinado por *keyspace* (banco de dados);
 - Um fator de replicação de 5, significa que 5 réplicas de cada linha serão gerenciadas pelo *cluster*;

Cláudio Lúcio

33

Cassandra

- Disponibilidade:
 - Existem alguns tipos de replicações:
 - Simples: utiliza o '*consistent hashing*' ;
 - Grupo: permite implementar o conceito de diferentes '*rack's*' ou múltiplos *data center's*; Arquivo: `cassandra-topology.properties`



Cassandra

- Consistência:
 - A consistência pode ser escolhida (por requisição): estrita ou eventual;
 - No processo de escrita as seguintes opções são válidas:
 - *Any* (Eventual): a escrita é bem sucedida se é escrita em qualquer nó disponível;
 - *One*: a escrita é bem sucedida se é escrita em um nó responsável pela linha;
 - *Quorum*: a escrita é bem sucedida se é escrita em um conjunto de nós da réplica($(fator\ de\ replicação /2)+1$);
 - *Local Quorum*: a escrita é bem sucedida se é escrita em um conjunto de nós da réplica no mesmo local ;
 - *All* (estrita):a escrita é bem sucedida se é escrita em todos os nós que contenham a chave;

Cláudio Lúcio

35

Cassandra

- Consistência:
 - Passos para o processo de escrita, exemplo:
 - Processo tenta fazer a escrita para todos os nós que contenham a linha;
 - Se todos os nós não estão disponíveis, dados são armazenados em um nó descrevendo todo o processo de escrita (espécie de log);
 - Se algum nó volta a ativa ou é contatado, a atualização é realizada;
 - Algumas opções de escrita estão disponíveis para leitura (*Any*, *One*, *Quorum*, *Local_Quorum* e *All*)

Cassandra

- Consistência:

- Processo de reparação para leitura:
 - Garante que dados acessados com elevada frequência estarão sempre consistentes;
 - No processo de leitura há uma comparação dos dados oriundos das réplicas: se existe inconsistência um processo de escrita é acionada para as demais réplicas;
 - O processo de reparação para leitura vem configurado como padrão
 - A configuração pode ser feita por família de colunas ;
- Exemplo de leitura e escrita:

```
SELECT total_purchases FROM SALES
USING CONSISTENCY QUORUM
WHERE customer_id = 5

UPDATE SALES
USING CONSISTENCY ONE
SET total_purchases = 500000
WHERE customer_id = 4
```

Cláudio Lúcio

37

Atividade e Reflexão

Atividade

- Demonstração utilizando o Cassandra
- Reflexão
 - Em quais cenários os bancos de dados baseados em colunas são recomendáveis?
 - Como é tratada a consistência no Cassandra?
 - Importância de ser avaliar o teorema CAP em uma solução NoSQL?

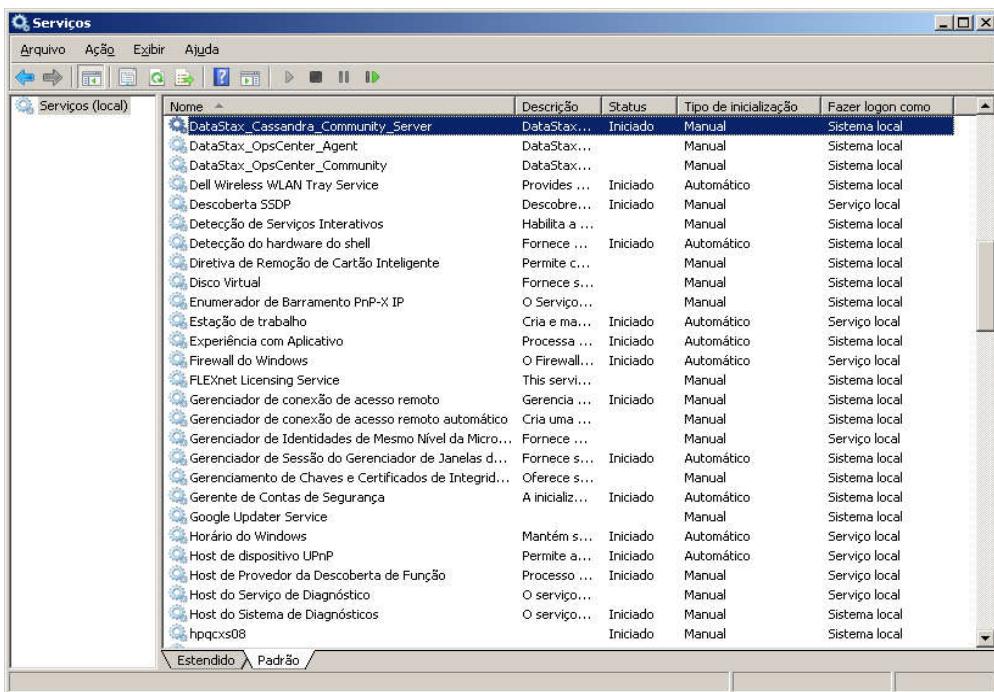


1) Instalação do DataStax Cassandra:

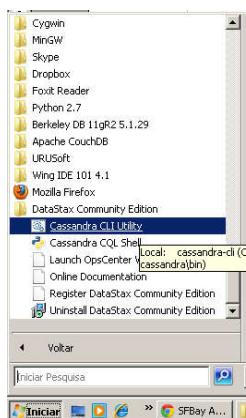
Faça download e instale o produto:

<http://downloads.datastax.com/community/datastax-community-32bit.msi>
<http://downloads.datastax.com/community/datastax-community-64bit.msi>

2) Verifique o serviço:



3) Acesse o Cassandra Cli Utility:



4) Conectando no servidor:

Digite: connect localhost /9160;

5) Criação de um banco de dados

```
1- CREATE KEYSPACE pucdemo  
with placement_strategy = 'org.apache.cassandra.locator.SimpleStrategy'  
and strategy_options = {replication_factor:1};
```

```
2- use pucdemo;
```

3- Criação de uma família de colunas - estática

```
CREATE COLUMN FAMILY aluno  
WITH comparator = UTF8Type  
AND key_validation_class=UTF8Type  
AND column_metadata = [  
    {column_name: nome, validation_class: UTF8Type}  
    {column_name: email, validation_class: UTF8Type}  
    {column_name: estado_civil, validation_class: UTF8Type}  
    {column_name: sexo, validation_class: UTF8Type}  
    {column_name: idade, validation_class: LongType}];
```

4- Criação de uma família de colunas - dinâmica

```
CREATE COLUMN FAMILY notas  
WITH comparator = UTF8Type  
AND key_validation_class=UTF8Type  
AND default_validation_class = LongType;
```

6) Acessando o banco de dados – API Cassandra

1- Inserindo dados

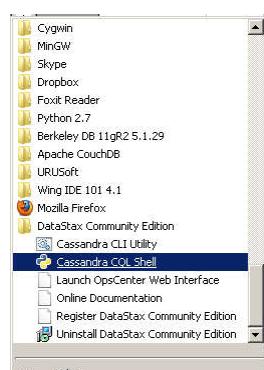
```
SET aluno['aluno1']['nome']='nome do aluno1';  
SET aluno['aluno1']['email']='meu_email@noemail.com';  
SET aluno['aluno1']['estado_civil']='solteiro';  
SET aluno['aluno1']['sexo']='M';  
SET aluno['aluno1']['idade']=25;  
SET aluno['aluno2']['nome']='nome do aluno2';  
SET aluno['aluno2']['email']='email@noemail2.com';  
SET aluno['aluno2']['estado_civil']='casado';  
SET aluno['aluno2']['sexo']='F';  
SET aluno['aluno2']['idade']=37;  
  
SET notas['aluno1']['materia 1'] = '9';  
SET notas['aluno2']['materia 1'] = '8';  
SET notas['aluno1']['Top. Especiais BD 1'] = '10';  
SET notas['aluno2']['Top. Especiais BD 1'] = '10';  
SET notas['aluno1']['SQL/MDX'] = '10';  
SET notas['aluno2']['SQL/MDX'] = '10';
```

2- Consultando dados

```
get aluno['aluno1'];  
get aluno['aluno2'];  
get notas['aluno2'];  
get notas['aluno2'];
```

7) Acessando o banco de dados – CQL

Clique na opção 'Cassandra CQL Shell'



```
select * from pucdemo.aluno;  
select * from pucdemo.notas;
```

```
update pucdemo.notas USING CONSISTENCY ALL set 'SQL/MDX'=9 where key='aluno1';
select * from pucdemo.notas;
```

8) Limitações do CQL

Não suporta operações de GROUP BY, ORDER BY (incluído no CQL 3.0);



Introdução ao Hadoop

Cláudio Lúcio

1

Introdução ao Hadoop

Agenda

- Conceitos
- Hive



Introdução ao Hadoop

Conceitos

Cláudio Lúcio

3

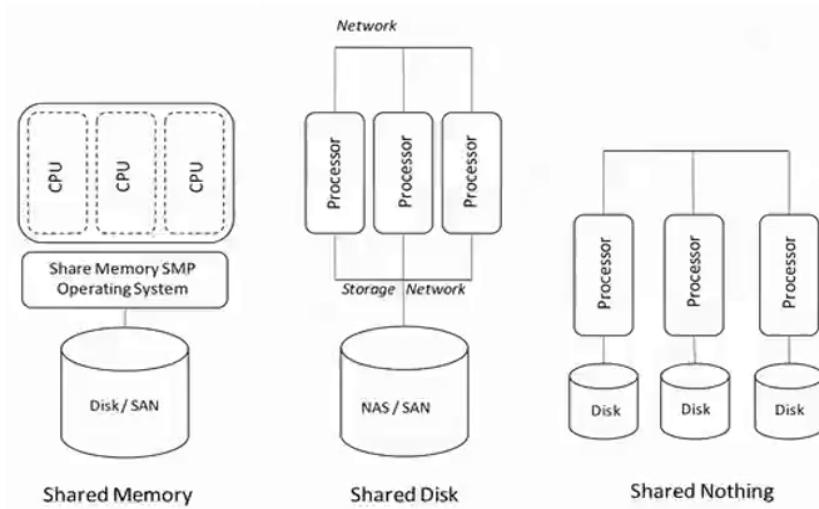
Hadoop - Introdução

Motivação:

- Estrutura de arquivos convencional não é capaz de lidar com dados massivos;
- É necessário um paradigma que comporte escalabilidade elástica;
- Bancos de dados relacionais utilizam conceitos, que às vezes, não são um requisito para dados massivos;
- Um infra estrutura tolerante a falhas e que permita computação paralela é necessária;

Hadoop - Introdução

Mudança arquitetural:



Cláudio Lúcio

Hadoop - Introdução

Mudança arquitetural:

Requisitos:

- Demandar estrutura de computação paralela;
- Esquemas de particionamentos (*shared nothing*);
- Tolerância a falhas;
- Escalabilidade horizontal e elástica;

Hadoop - Introdução

Mudança arquitetural:

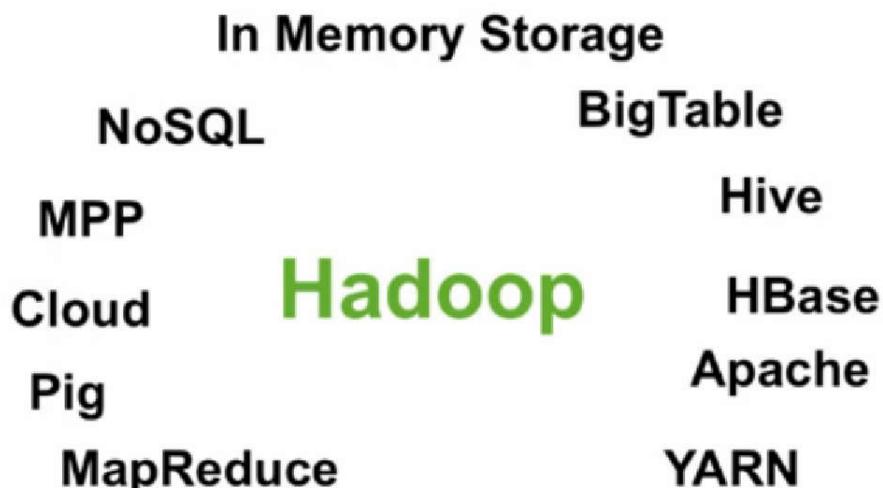
Alguns paradigmas de computação são fundamentais:

- Várias soluções vão utilizar sistemas de arquivos distribuídos;
- Esquemas de controle de réplicas e consistência dos dados;
- Algoritmos para execução paralela em conjunto com estruturas de arquivos;

Cláudio Lúcio

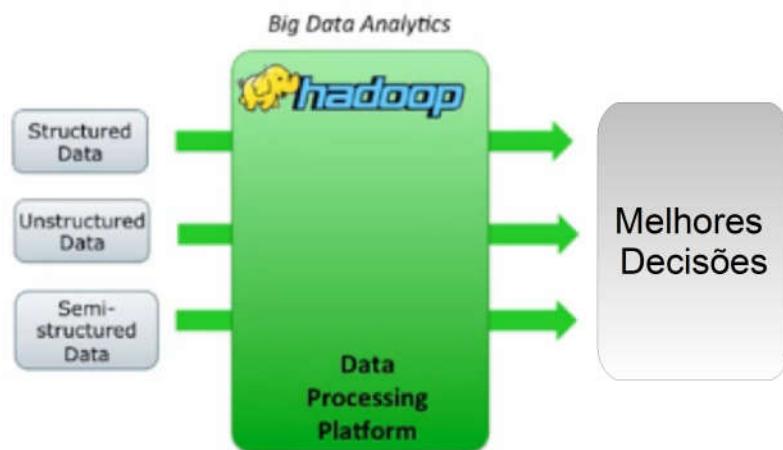
Hadoop - Introdução

Visão Geral



Hadoop - Introdução

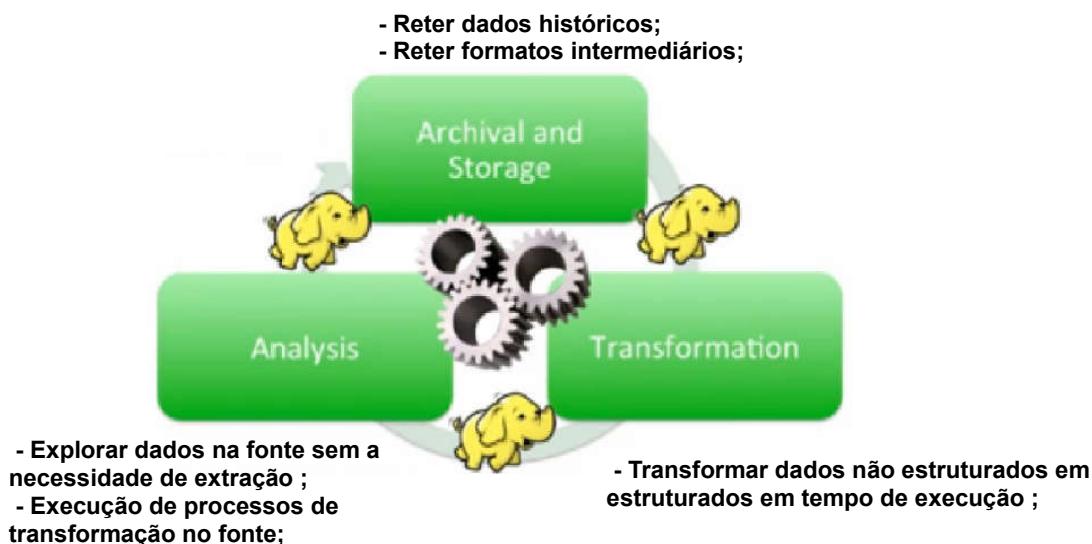
Visão Geral



Cláudio Lúcio

Hadoop - Introdução

Visão Geral



Cláudio Lúcio

Hadoop - Introdução

Hadoop

- Plataforma para acesso a dados estruturados, semi estruturados(logs, tweets, sensor data) e não estruturados;
- Executar o ciclo de: obter, reter e analisar grandes volumes de dados;
- *Open Source* e mantida pela fundação Apache;
- Características:
 - Escalabilidade;
 - Replicação de dados distribuída;
 - Utilização de '*commodity hardware*' (NOW);

Cláudio Lúcio

Hadoop - Introdução

Hadoop versus SGBDR

Características	SGBDR	HADOOP
Esquema	Esquema na escrita	Esquema na leitura
Desempenho	Leituras são rápidas	Escrita é rápida
Governança	Dados estruturados e padrões	Não estruturado
Processamento	Limitado	Ilimitado
Tipos de dados	Estruturado	Não estruturado
Escalabilidade	Vertical, em casos especiais é horizontal	Elástica
Exemplo de uso	- Transações ACID - <i>Operational Data Store</i> - Acesso a dados interativo	- <i>Data Discovery</i> - Armazenamento e processamento de dados massivos

Hadoop - Introdução

Hadoop Componentes – Core

Hadoop: Plataforma para processamento e armazenamento de dados massivos (larga escala)

HDFS



- Sistema de arquivos distribuídos para dados estruturados, semi-estruturados e não estruturados. Arquivos são divididos em bloco e armazenados com redundância no cluster;

Map Reduce

- Framework para execução de processamento paralelos em múltiplos nós de trabalho para posteriormente combinar os resultados;

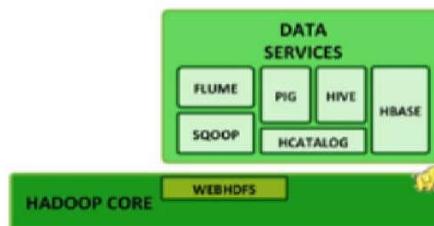
YARN(Hadoop 2.0)

- Gerenciamento da execução das aplicações no cluster;

Cláudio Lúcio

Hadoop - Introdução

Hadoop Componentes – Data Services



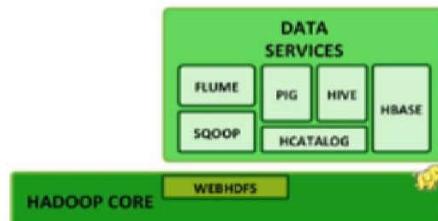
WEBHDFS

- API REST que permite acesso via HTTP ao HDFS: movimentação de arquivos(entrada e saída) exclusão de arquivos;
- Executar funções de arquivos e diretórios;
- `webhdfs://<host>:<http port>/path`

Hadoop - Introdução

Hadoop Componentes – Data Services

Flume



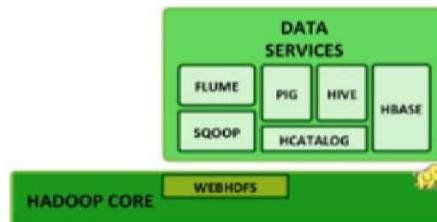
- Serviço distribuído para coleta, agregação e movimentação de *streams* de logs de dados para o HDFS;
- Executar funções de *streaming* com capacidade de recuperação e tolerante a falhas;
- Uso principal é para movimentação de arquivos de log diretamente para o HDFS/Hadoop;

Cláudio Lúcio

Hadoop - Introdução

Hadoop Componentes – Data Services

Sqoop



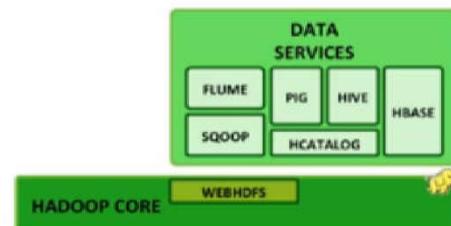
- Movimentação de dados para dentro do Hadoop a partir de bancos de dados relacionais e vice versa;
- Ferramentas e conectores que permitem dados de bancos de dados relacionais(Oracle, DB2, MySQL) serem armazenados e obtidos do Hadoop;

Cláudio Lúcio

Hadoop - Introdução

Hadoop Componentes – Data Services

PIG



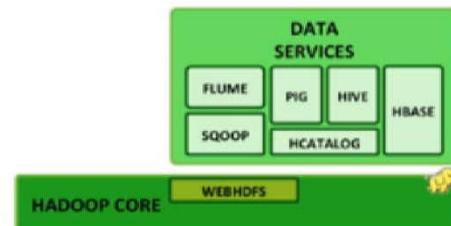
- Possibilitar a escrita de programas de transformação e movimentação de dados utilizando uma linguagem simples de script;
- Pig Latin é a linguagem que define um conjunto de transformações nos dados permitindo: junção, agregação, ordenação entre outros;
- A linguagem também pode ser estendida utilizando UDF's que podem ser escritas em JAVA e executadas a partir de comando Pig latin;

Cláudio Lúcio

Hadoop - Introdução

Hadoop Componentes – Data Services

Hive



- Interface SQL para o Hadoop que possibilita summarização de dados, consultas ad-hoc e análise de grandes volumes de dados;
- Existem conectores que utilizam Hive e permitem conexões de algumas ferramentas de BI: Microstrategy, Excel, PowerPivot, Tableau e outros;
- HiveQL (HQL) é a linguagem utilizada no Hive: 'compatibilidade' com o SQL;

Cláudio Lúcio

Hadoop - Introdução

Hadoop Componentes – Data Services



HCatalog

- Serviço de metadados que permite usuários acessarem dados no hadoop como um conjunto de tabelas sem a necessidade de fornecer detalhes sobre onde e como os arquivos estão armazenados;
- Possibilita compartilhamento e interoperabilidade entre outras ferramentas de data service:Pig, Map Reduce e Hive;
- Permite também interoperabilidade e acesso de dados para outros bancos de dados como SQL Server e Teradata(conexão hadoop via HCatalog);

Cláudio Lúcio

Hadoop - Introdução

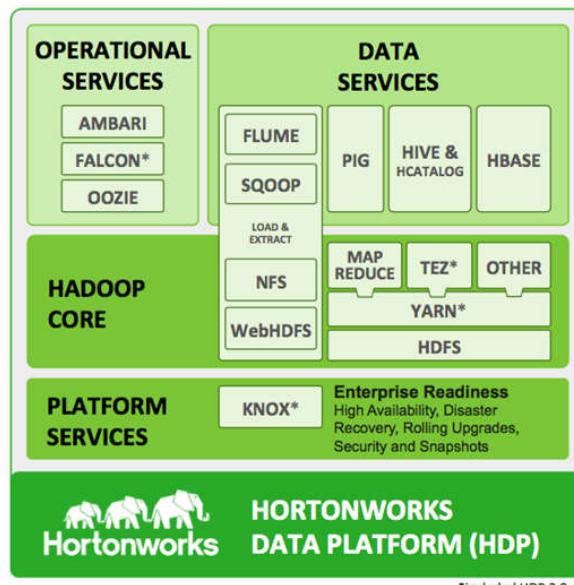
Hadoop Componentes – Data Services

HBASE

- Banco de dados NoSQL(colunar -'*big table clone*') para utilização de dados de forma interativa (não batch) ;
- Comumente utilizando para servir aplicações inteligentes: recomendações de produtos, predição de comportamento de usuários;

Hadoop - Introdução

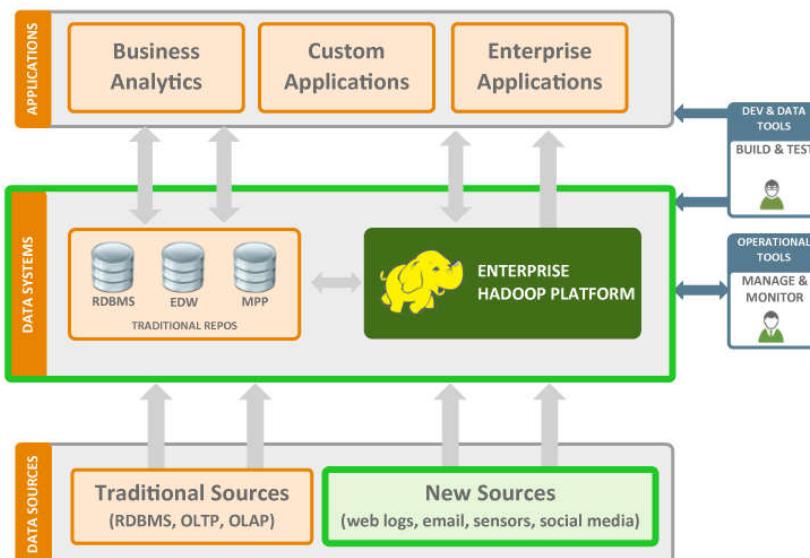
Hadoop - Hortonworks



Cláudio Lúcio

Hadoop - Introdução

Hadoop - Hortonworks – BI Approach



Hadoop - Introdução

Atividade

Utilizando a máquina virtual da HortonWorks

Cláudio Lúcio

NoSQL
MongoDB
MapReduce
Cassandra
Python
Processamento
BerkeleyDB
Paralelismo
chave-Valor
Documentos
DFS
Colunar

Hive

Conceitos

Hive - Conceitos

Hive

- Software para '*data warehouse*' que facilita a consulta e gerenciamento de grandes massas de dados residentes em sistema de armazenamento distribuído;
- Características Hive:
 - Ferramentas capazes de realizar processos de ETL;
 - Mecanismo que cria uma estrutura para arquivos (independente do formato);
 - Acesso a arquivos residentes no HDFS ou outras fontes de dados, HBase por exemplo;
 - Execução de consultas utilizando HQL ou via MapReduce

Cláudio Lúcio

Hive - Conceitos

Hive

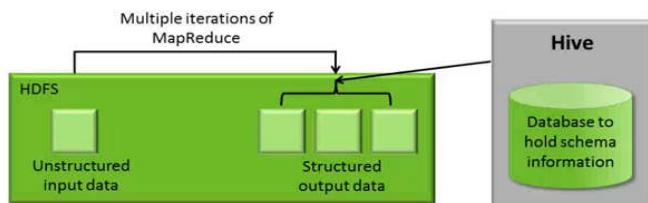
- 'Estruturar' dados não estruturados;
- Acessar este dados utilizando uma ferramenta '*SQL-Like*': HiveQL- HQL;
- Algumas verdades:
 - Não é um banco de dados relacional. Dados estão no HDFS;
 - Não foi desenvolvido para para processamentos *on-line*. Consultas executam no Hadoop: Map Reduce. Pode haver latência elevada;
 - Não é recomendado para consultas de tempo real ou para fazer atualizações nos dados;



Hive - Conceitos

Hive

Caso de uso:



Você precisa:

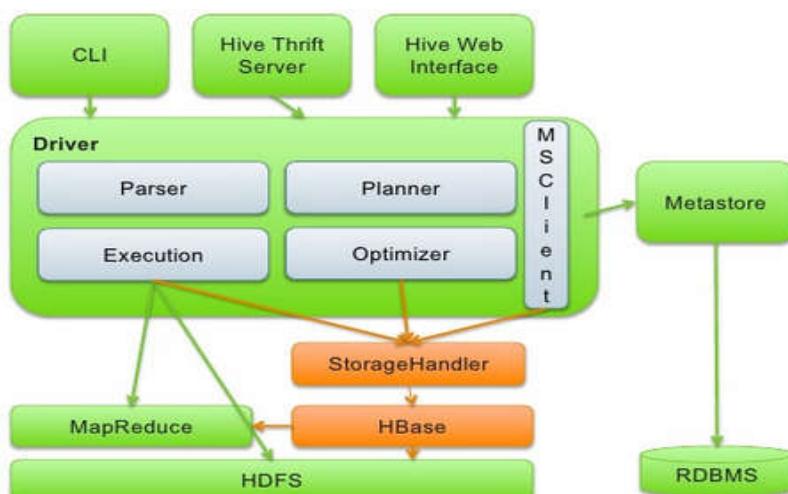
- Fazer consultas '*ad-hoc*';
- Sumarizar os dados;
- Fazer análise dos dados;

Cláudio Lúcio

Hive - Conceitos

Hive

Arquitetura Hive:



Hive - Conceitos

HCatalog

- Era um projeto independente, mas foi fundido com o Hive em 26/03/2013;
- Criar e prover acesso aos metadados Hive para ferramentas internas ou externas no Hadoop;
- Características HCatalog:
 - Prover metadados(esquema e tipo de dados) para ferramentas hadoop;
 - Prover a abstração de 'tabelas' – não há necessidade de saber onde e como as 'tabelas' armazenadas;
 - Interoperabilidade: Pig, Map Reduce e Hive;
 - Oferecer interface de acesso REST para metadados Hive;

Cláudio Lúcio

Hive - Conceitos

HCatalog

- Apresentar para usuários uma 'visão relacional' dos dados;
- Por padrão suporta arquivos: CSV, JSON e arquivos sequenciais;
- Pode ser estendido uma vez que seja fornecido classes de input e output;
- Tabelas podem ser particionadas por uma ou mais chaves (utilizando *hash*);
- Para tabelas particionadas não garante consistência de leitura (no caso da exclusão de uma partição);

Cláudio Lúcio

Hive - Conceitos

HCatalog

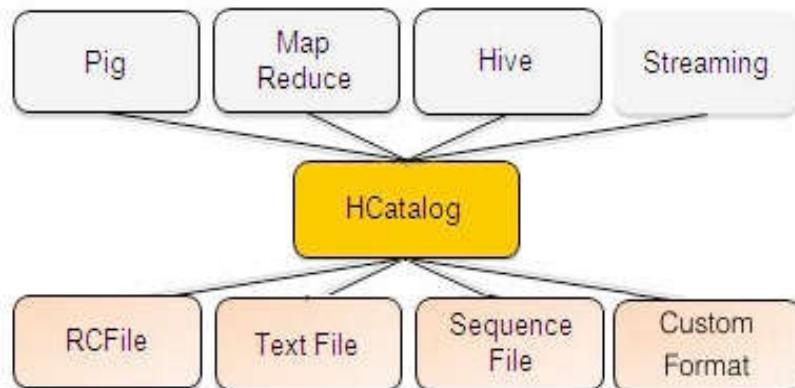
- Uma vez que uma partição é criada, dados não podem ser inseridos, atualizados ou removidos;
- Estrutura das partições é multidimensional;

Cláudio Lúcio

Hive - Conceitos

HCatalog

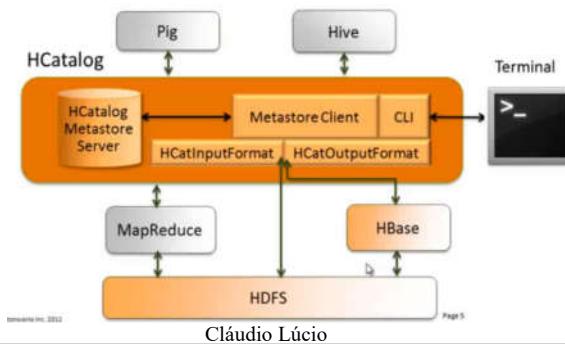
- Esquema de acesso via HCatalog



Hive - Conceitos

HCatalog

- Cria metadados para objetos e *Hive* (*HCatalog MetaStore*) a adicionamente fornece:
 - Para usuários *Pig*: acesso ao *HCatLoader* e *HCatStorer*,
 - Para usuários *Map Reduce*: acesso *HCatInputFormat* e *HCatOutputFormat*,



Hive - Conceitos

HCatalog

- Exemplos de script HCatalog:

```
hcat.py -f meuscript.hcatalog  
hcat.py -e "create table alunos (nota int)"
```

- Hcatalog DDL:

- CREATE/ALTER/DROP table;
- SHOW table;
- DESCRIBE;
- Suporte comandos DDL Hive;

Hive - Conceitos

HCatalog

- HCatalog exemplo DDL:

```
1   create table mytable (
      id          int,
      firstname   string,
      lastname    string
)
comment 'An example of an HCatalog table'
partitioned by (birthday string)
stored as sequencefile;
```

```
2   > hcat.py -f create_table.hcatalog
> hcat.py -e "describe mytable"
OK
id int
firstname string
lastname string
birthday string
Time taken: 0.859 seconds
```

Cláudio Lúcio



Hive

SQL e HQL

Hive – SQL e HQL

SQL

- Grande número de implementações para os SGBDR: mercado sedimentado;
- Padrão de mercado: muitas aplicações utilizam SQL;
- Primeira publicação ISO foi em 1987. Outras versões revisadas aconteceram em 1989, 1992, 1999, 2003, 2008 e 2011;
- Muitas pesquisas indicam SQL como um linguagem fundamental para os cientistas de dados;

Cláudio Lúcio

Hive – SQL e HQL

SQL – Breve Revisão

Data Manipulation Language – DML

```
SELECT ... FROM ... WHERE ...
INSERT INTO ... VALUES ...
UPDATE ... SET ... WHERE ...
DELETE FROM ... WHERE ...
```

- A Cláusula SELECT:

```
SELECT      [DISTINCT] {*| column [alias], ...}
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

Hive – SQL e HQL

Hive

HiveQL - HQL

- Baseado na especificação SQL-92;
- Converte SQL em processo *MapReduce*;
 - 'Inserts' são executados de forma paralela, por exemplo;
- Permite também a criação de códigos específicos de *MapReduce*;
- Também é possível criar UDF que podem ser executadas nos comandos HQL;
- No Hadoop 2.0 com o uso do Yarn a '*Hortonworks*' reformulou o Hive para processamento em tempo real sem o uso de mapreduce;

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Tabelas Hive
- Consistem de:
 - Dados: arquivo ou conjunto de arquivos no HDFS, por exemplo;
 - Esquema(schema): metadados armazenados no *metastore*;
- Esquema e dados são separados:
 - Um esquema pode ser definido para dados já existentes;
 - Dados podem ser adicionados ou removidos de forma independentes do esquema;
 - É necessário utilizar um esquema no Hive para 'enxergar' os dados;

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Exemplo de criação

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
    page_url STRING, referrer_url STRING,
    ip STRING COMMENT 'IP Address of the User')
COMMENT 'This is the page view table'
PARTITIONED BY(dt STRING, country STRING)
STORED AS SEQUENCEFILE;
```

Cláudio Lúcio

Hive – SQL e HQL

Hive

Exemplos HQL

```
hive> SHOW TABLES
hive> DESCRIBE mytable;
hive> ALTER TABLE mytable
RENAME to mt;
hive> ALTER TABLE mytable ADD
COLUMNS (mycol STRING);
hive> ALTER TABLE mytable DROP
PARTITION (age=17)
```

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Carregando dados:

- Utilizar o comando LOAD DATA para importar arquivos:

```
hive>LOAD DATA LOCAL INPATH 'input/meudados/dados.txt' INTO TABLE MTABELA;
```
- Arquivos não são alterados pelo comando;
- Hive armazena os dados em um diretório padrão:
/hive/warehouse;
- Há o comando OVERWRITE para substituir dados já existentes;
- O esquema da tabela é avaliado quando os dados são lidos e não quando são gravados; Se a linha não esta consistente com o esquema a linha é lida como nula;

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Carregando dados:

- Utilizar o comando INSERT :

```
hive>CREATE TABLE idade_emp(nom_id string,idade int);
hive>INSERT OVERWRITE TABLE idade_emp select
idade,count(idade) from empregados;
```

- Utilização do comando para inserir dados de uma tabela em outra;
- Normalmente resultados de uma consulta de Big Data são muito grandes, alternativa: utilizar o comando *insert* para reter os dados da consulta;
- OVERWRITE: executa um comando *delete* na tabela, caso contrário será realizado um *append*;

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Particionamentos

- Hive organiza tabelas em partições
- As tabelas são divididas em '*buckets*' de acordo com o(s) domínio(s) de uma ou mais colunas;
- As partições são baseadas em um função *hash* que é aplicada na(s) coluna(s);
- O tempo de resposta das consultas pode ser otimizado(cláusulas baseadas em *where*), cada valor da coluna é armazenado em um diretório específico;

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Particionamentos

- Para agrupamentos pode impactar negativamente o desempenho;
- Para casos em que o domínio é muito grande, pode causar um overhead para o NameNode (mantém todo sistema de arquivo em memória – centenas de milhares de arquivos, por exemplo);

Hive – SQL e HQL

Hive

- Particionamentos

- Exemplos:

```
- CREATE TABLE partitioned_user (firstname VARCHAR(64), lastname VARCHAR(64), address
    STRING, city    VARCHAR(64), post     STRING, phone1   VARCHAR(64), phone2
    STRING, email    STRING, web      STRING )
    PARTITIONED BY (country VARCHAR(64), state VARCHAR(64)) STORED AS SEQUENCEFILE;

- LOAD DATA LOCAL INPATH '${env:HOME}/staticinput.txt'
  INTO TABLE partitioned_user
  PARTITION (country = 'US', state = 'CA');

- Será criado a seguinte estrutura:
  /user/hive/warehouse/partitioned_user/country=US/state=CA/
```

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Consultas:

- HQL suporta:

- Cláusula *Where*;
- Comandos *Union ALL* e *Distinct*,
- *GROUP BY* e *HAVING*;
- *JOINS*
- Cláusula *LIMIT*: limitar o número de linhas de forma aleatória;
- Sub consultas

- Somente na cláusula *FROM*

```
SELECT total FROM
  (SELECT c1 + c2 AS total FROM
  mytable) my_query;
```

Nomear a sub-consulta

Hive – SQL e HQL

Hive

- Consultas - exemplos:

```
• SELECT * FROM customers;
• SELECT COUNT(1) FROM customers;
• SELECT firstName, lastName, address,
zip FROM customers WHERE orderID > 0
GROUP BY zip;
• SELECT customers.* , orders.* FROM
customers JOIN orders ON
(customers.customerID =
orders.customerID);
• SELECT customers.* , orders.* FROM
customers LEFT OUTER JOIN orders ON
(customers.customerID =
orders.customerID);
```

Cláudio Lúcio

Hive – SQL e HQL

Hive

- Funções:

- Matemáticas

- round(), floor(), ceil(), rand() ...

- Caractere

- concat(), substr(), upper(), lower(), regexp_replace(), get_json_object() ...

- Data

- year(), month(), day(), rand() ...

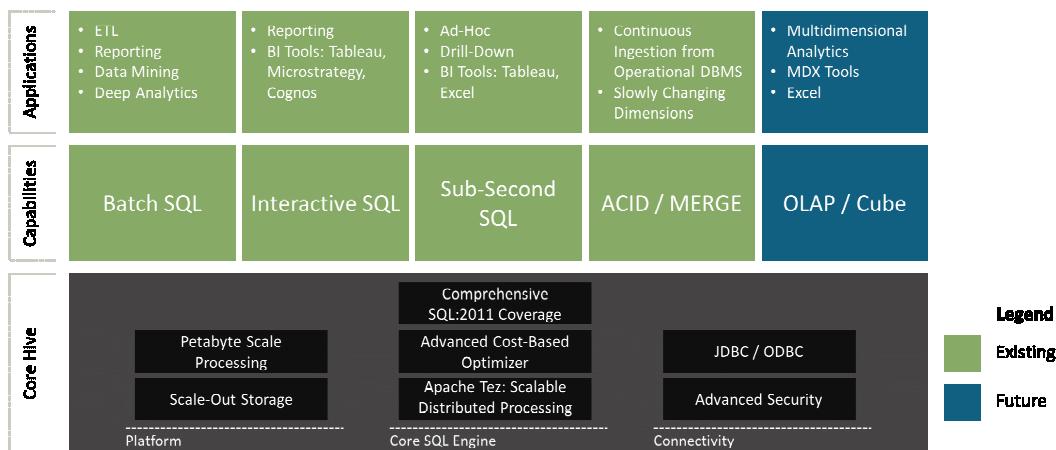
- Estatística descritiva

- count(), avg(), sum(), max(), min() ...

Hive – SQL e HQL

Hive

Visão geral do projeto – final de 2016



Cláudio Lúcio

Hive – SQL e HQL

Hive e Tez

TEZ



- É um framework para execução de aplicações baseadas em dados de forma distribuída;
- Computação é expressa em um fluxo de dados → grafo direcionado acíclico;
- Totalmente integrado no YARN;
- Possui várias esquemas de processamento para execução dos grafos;

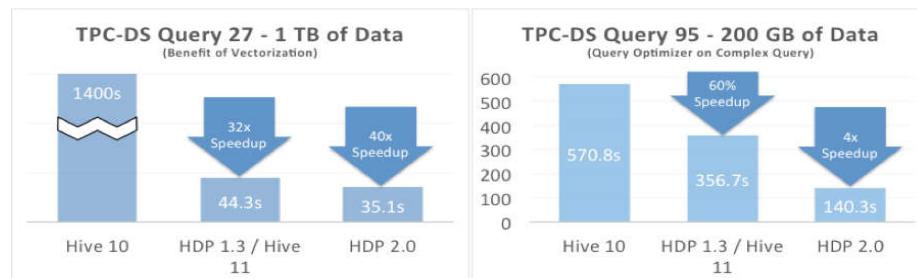
Hive – SQL e HQL

Hive e Tez

Stinger - HortonWorks



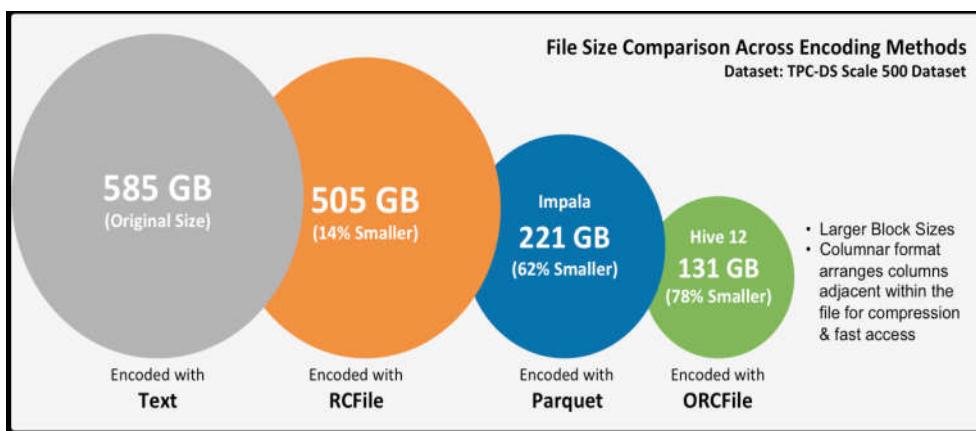
- Projeto da HortonWorks para fazer com o Hive executa-se utilizando o TEZ;
- A meta do projeto era fazer com algumas consultas melhorassem 100 X.



Cláudio Lúcio

Hive – SQL e HQL

Hive e Tez



Hive

Atividade

Utilizando o Hive para análises de Big Data

Cláudio Lúcio

Atividade e Reflexão

Atividade

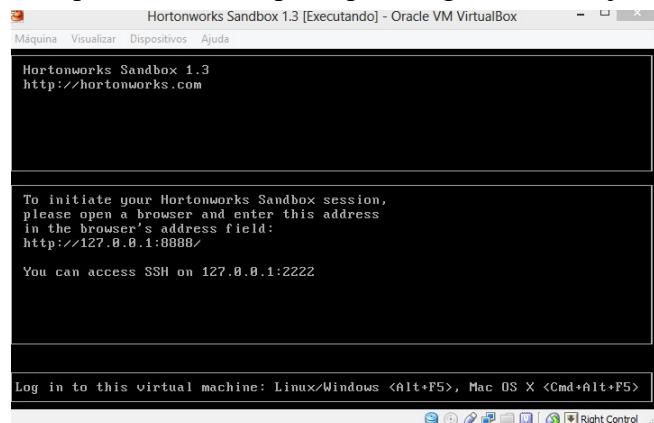
- Demonstração utilizando o Hive
- Reflexão
 - Como você entende/descreveria a plataforma Hadoop?
 - Quais são seus principais componentes (fundamentais)?
 - Qual a função do Hive para a plataforma Hadoop? Quais seriam seus casos de uso, pensado em uma arquitetura distribuída de dados?
 - Você imagina algum caso prático em que poderia usar o Hive?

- 1) Para utilização das práticas com o Hadoop vamos usar uma sandbox (máquina virtual) fornecida pela HortonWorks. Esta máquina virtual possui os seguintes produtos instalados:

Esta é a versão 2.1 do produto e pode ser obtida em:

https://hortonassets.s3.amazonaws.com/2.1/virtualbox/Hortonworks_Sandbox_2.1.ova

- 2) Inicialize a máquina virtual. Espere que a seguinte tela seja exibida:



- 3) Pressione Alt+F5 e faça o login com o usuário *root* e senha indicada:*hadoop*;
- 4) Vamos alterar para um usuário administrativo do hadoop

```
su hdfs  
cd /home/hdfs/
```

- 5) Para listar o conteúdo dos arquivos:

```
hadoop fs -ls /
```

- 6) Inicialmente vamos fazer a criação de uma estrutura de diretório no cluster:

```
hadoop fs -mkdir /dados  
hadoop fs -ls /  
hadoop fs -ls /dados  
hadoop fs -mkdir /dados/bigdata  
hadoop fs -ls /dados
```

- 7) Testando a exclusão de um diretório:

```
hadoop fs -rmr /dados/bigdata  
hadoop fs -ls /dados  
hadoop fs -mkdir /dados/bigdata  
hadoop fs -ls /dados
```

- 8) Adicionando um arquivo externo para o cluster:

```
ls  
hadoop fs -ls /dados/bigdata  
hadoop fs -put /var/log/boot.log /dados/bigdata  
hadoop fs -ls /dados/bigdata
```

- 9) Copiando arquivos no cluster:

```
hadoop fs -ls /apps/hive/warehouse/sample_07  
hadoop fs -ls /dados/bigdata  
hadoop fs -cp /apps/hive/warehouse/sample_07/sample_07 /dados/bigdata  
hadoop fs -ls /dados/bigdata
```

10) Obtendo arquivos do cluster:

```
ls  
hadoop fs -ls /dados/bigdata  
hadoop fs -get /dados/bigdata/sample_07 /home/hdfs/  
cd root  
ls  
rm sample_07  
ls
```

11) Listando o conteúdo de um arquivo:

```
hadoop fs -cat /dados/bigdata/sample_07
```

12) HUE: Outra forma de acesso:

Abra um navegador e execute a url <http://localhost:8000> ;

Clicando no ultimo ícone a seguinte tela será apresentada:

The screenshot shows the Hortonworks Sandbox 1.3 interface. At the top, there's a navigation bar with links like 'Beeswax (Hive UI)', 'Pig', 'HCatalog', 'File Browser', 'Job Browser', 'Job Designer', and 'Oozie Editor/Dashboard'. Below the navigation bar is a 'Hue Architecture' diagram. The diagram illustrates the flow of data between the 'Hue UI' (a computer monitor icon), 'Hue Server' (a central box with a 'hue' logo), 'Hue Apps' (represented by various application icons like Oozie, Beeswax, YARN, JobTracker, Hue Plugins, HDFS, HBase, and Pig), and the 'Hue DB' (a cylinder icon). Arrows indicate the communication paths between these components, labeled 'Thrift/REST'. On the left side of the interface, there's an 'Introduction' section with text about the Hortonworks Sandbox and its features. At the bottom, there's a toolbar with icons for 'VM', 'Screenshots', 'Hortonworks San...', 'Network and Shar...', 'Oracle VM Virtual...', 'Hortonworks San...', 'Desktop', and a date/time stamp '22/01 29/08/2013'.

Esta ferramenta se chama HUE e permite interações com o cluster hadoop. Inclue várias ferramentas de fácil utilização, dentre elas: Pig, Hive, Hbase, Jobs Map Reduce, dentre outros.

As aplicações HUE são totalmente Web, permitindo que se faça execute as aplicações de acordo com os botões abaixo:



13) No caso de selecionarmos o quinto botão ele exibe uma aplicação que o “File Browser” . As atividades realizadas anteriormente via linha de comando podem ser feitas utilizando esta interface.

14) Navegue na estrutura e veja o arquivo anteriormente carregado;

15) Exclua o arquivo start_ambari.sh utilizando o botão “Delete”;

16) Faça também a exclusão do arquivo sample_07.csv;

17) Acesse o seguinte caminho: /apps/hive/warehouse/ ;

18) Copie o arquivo sample_07 utilizando o botão “Copy”(não se esqueça de marcar o arquivo antes de clicar no botão);

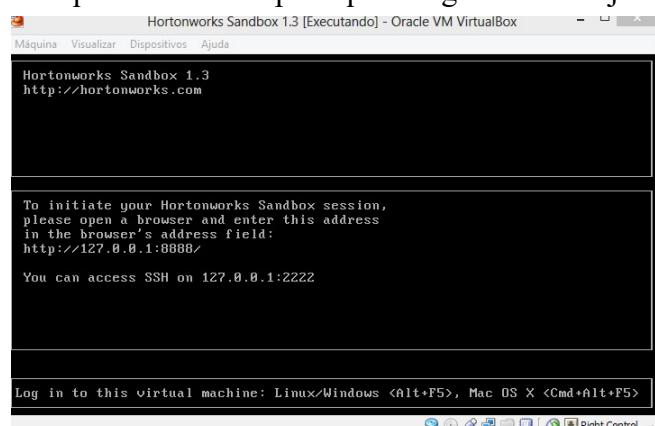
19) Para o novo destino indique o caminho: /dados/bigdata

- Para utilização das práticas com o Hadoop vamos usar uma sandbox (máquina virtual) fornecida pela HortonWorks. Esta máquina virtual possui os seguintes produtos instalados:

Apache Hadoop	1.2.0
Apache Hive includes HCatalog	0.11.0
Apache HBase	0.94.6.1
Apache ZooKeeper	3.4.5
Apache Pig	0.11
Apache Sqoop	1.4.3
Apache Oozie	3.3.2
Apache Ambari	1.2.4
Apache Flume	1.3.1
Apache Mahout	0.7.0

Esta é a versão 2.3.2 do produto e pode ser obtida em:
https://hortonassets.s3.amazonaws.com/2.1/virtualbox/Hortonworks_Sandbox_2.1.ova

- Inicialize a máquina virtual. Espere que a seguinte tela seja exibida:



- Abra um navegador e execute a url <http://localhost:8000> ;
- Para este tutorial será utilizado uma base de dados com estatísticas de jogos de baseball entre os anos 1871 até 2011. Este arquivo possui 95.000 linhas. A base de dados pode ser acessada em: <http://seanlahman.com/files/database/lahman591-csv.zip> .
- Apenas dois arquivos serão utilizados: master.csv e batting.csv
- Fazendo Upload dos arquivos:
 Selecione a opção 'File Browser' no topo da página



- Navegue na estrutura para o root e entre na pasta /data;
- Clique no botão *upload* e selecione a opção *files*

File Browser

The screenshot shows the HDFS File Browser interface. At the top, there are buttons for 'Search for file name', 'Rename', 'Move', 'Copy', 'New', 'Upload', 'Change Permissions', 'Download', 'Delete', and 'Zip file'. Below the toolbar, the path is shown as 'Home / user / sandbox'. On the right, there is a 'Trash' button. The main area displays a list of files in the 'sandbox' directory, including '.', '..', '.Trash', 'Batting.csv', and 'Master.csv'.

- Selecione os dois arquivos: master.csv e batting.csv;
- Veja os arquivos já no *cluster*:

File Browser

This screenshot shows the same File Browser interface as above, but with a different set of files listed in the 'sandbox' directory. The visible files are '.', '..', '.Trash', 'Batting.csv', and 'Master.csv', each with its size, user, group, permissions, and last modified date.

7) Criando as tabelas utilizando HCatalog

- Com arquivos já carregados no HDFS, agora serão criadas as tabelas utilizando o HCatalog/Hive (metastore). Selecione a opção HCat no menu:



- No menu 'Actions' selecione a opção 'Create a new table from a file';

Create a new table from a file

The screenshot shows the 'Create a new table from a file' dialog. It has two main sections: 'Table options' and 'File options'. In 'Table options', there is a 'Table Name' field with 'table_name' and a 'Description' field with 'Optional'. In 'File options', there is an 'Input File' field with '/user/user_name/data_dir' and a 'Choose a file' button. At the bottom is a 'Create table' button.

- Criar uma tabela para o arquivo batting.csv:
 - nome da tabela: batting_data;
 - O item 'optional' deixe em branco;
 - Selecione o arquivo;
 - Veja que algumas opções já estarão marcadas, por padrão. E o arquivo será pré exibido;

Create a new table from a file

ACTIONS Create a new table from a file Create a new table manually	Table options Table Name: <input type="text" value="batting_data"/> Description: <input type="text" value="Optional"/>																																																				
File options Input File: <input type="text" value="/user/sandbox/Batting.csv"/> Choose a file Encoding: <input type="text" value="Unicode UTF8"/> <input checked="" type="checkbox"/> Read column headers <input checked="" type="checkbox"/> Import data Delimiter: <input type="text" value="Comma (,)"/> <input checked="" type="checkbox"/> Autodetect delimiter <input type="checkbox"/> Ignore whitespaces Replace delimiter with: <input type="text" value="^A' (\001)"/> <input type="checkbox"/> Java-style comments <input type="checkbox"/> Ignore tabs Single line comment: <input type="text"/>																																																					
Table preview <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Column name</th> <th>Column name</th> <th>Column name</th> <th>Column name</th> </tr> <tr> <th>playerid</th> <th>yearid</th> <th>stint</th> <th>teamid</th> </tr> <tr> <th>Column type</th> <th>Column type</th> <th>Column type</th> <th>Column type</th> </tr> <tr> <th>string</th> <th>int</th> <th>int</th> <th>string</th> </tr> </thead> <tbody> <tr> <td>Row #1 aardsda01</td> <td>2004</td> <td>1</td> <td>SFN</td> </tr> <tr> <td>Row #2 aardsda01</td> <td>2006</td> <td>1</td> <td>CHN</td> </tr> <tr> <td>Row #3 aardsda01</td> <td>2007</td> <td>1</td> <td>CHA</td> </tr> <tr> <td>Row #4 aardsda01</td> <td>2008</td> <td>1</td> <td>BOS</td> </tr> <tr> <td>Row #5 aardsda01</td> <td>2009</td> <td>1</td> <td>SEA</td> </tr> <tr> <td>Row #6 aardsda01</td> <td>2010</td> <td>1</td> <td>SEA</td> </tr> <tr> <td>Row #7 aaronha01</td> <td>1954</td> <td>1</td> <td>ML1</td> </tr> <tr> <td>Row #8 aaronha01</td> <td>1955</td> <td>1</td> <td>ML1</td> </tr> <tr> <td>Row #9 aaronha01</td> <td>1956</td> <td>1</td> <td>ML1</td> </tr> </tbody> </table>		Column name	Column name	Column name	Column name	playerid	yearid	stint	teamid	Column type	Column type	Column type	Column type	string	int	int	string	Row #1 aardsda01	2004	1	SFN	Row #2 aardsda01	2006	1	CHN	Row #3 aardsda01	2007	1	CHA	Row #4 aardsda01	2008	1	BOS	Row #5 aardsda01	2009	1	SEA	Row #6 aardsda01	2010	1	SEA	Row #7 aaronha01	1954	1	ML1	Row #8 aaronha01	1955	1	ML1	Row #9 aaronha01	1956	1	ML1
Column name	Column name	Column name	Column name																																																		
playerid	yearid	stint	teamid																																																		
Column type	Column type	Column type	Column type																																																		
string	int	int	string																																																		
Row #1 aardsda01	2004	1	SFN																																																		
Row #2 aardsda01	2006	1	CHN																																																		
Row #3 aardsda01	2007	1	CHA																																																		
Row #4 aardsda01	2008	1	BOS																																																		
Row #5 aardsda01	2009	1	SEA																																																		
Row #6 aardsda01	2010	1	SEA																																																		
Row #7 aaronha01	1954	1	ML1																																																		
Row #8 aaronha01	1955	1	ML1																																																		
Row #9 aaronha01	1956	1	ML1																																																		
Create table																																																					

- Altere o nome da coluna 'r' para 'runs' e altere se o tipo de dado para 'int';
- Clique no botão 'create table';

- Criar uma tabela para o arquivo master.csv:

- nome da tabela: master_data;
- O item 'optional' deixe em branco;
- Selecione o arquivo;

- Veja as duas tabelas criadas no HCatalog:

HCatalog: Table List

ACTIONS Create a new table from file Create a new table manually	Table Name batting_data Browse Data master_data Browse Data sample_07 Browse Data sample_08 Browse Data
---	--

- Clique no botão 'Browse Data' para visualizar as tabelas criadas;

8) Fazendo algumas análise com o Hive

- Arquivos já carregados no HDFS;
- Metadados já criados para o Hive. Desta forma pode-se trabalhar emitindo-se comandos em HiveQL;
- Para emitir este comando será utilizado uma ferramenta chamada **Beeswax**: interface interativa para o Hive em que poderá ser digitado consultas e as mesmas serão avaliadas pelo Hive e transformadas em uma série de jobs *Map Reduce*;
- Para executar o Beeswax acesse o menu conforme figura abaixo:

- As consultas podem ser digitadas e para execução acione o botão 'Execute'. Não é possível digitar mais de uma consulta, separando-as por “;”, por exemplo.
- Como HCatalog e Hive são integrados, nas verdade o mesmo produto, tudo que foi criado no HCatalog é válido para o Hive;
- Digite o comando para visualizar as tabelas:

```
show tables
```

tab_name
batting_data
master_data
sample_07
sample_08

- Hive utiliza o esquema/banco de dados criado no HCatalog. Para estes caso,não é necessário utilizar o nome do esquema. Digite o comando;

```
Select * from batting_data
```

	playerid	yearid	stint	teamid	lgid	g	g_batting	ab	runs	h	2b	3b	hr	rbi
0	aardsda01	2004	1	SFN	NL	11	11	0	0	0	0	0	0	0
1	aardsda01	2006	1	CHN	NL	45	43	2	0	0	0	0	0	0
2	aardsda01	2007	1	CHA	AL	25	2	0	0	0	0	0	0	0
3	aardsda01	2008	1	BOS	AL	47	5	1	0	0	0	0	0	0
4	aardsda01	2009	1	SEA	AL	73	3	0	0	0	0	0	0	0
5	aardsda01	2010	1	SEA	AL	53	4	0	0	0	0	0	0	0
6	aaronha01	1954	1	ML1	NL	122	122	468	58	131	27	6	13	69
7	aaronha01	1955	1	ML1	NL	153	153	602	105	189	37	9	27	106
8	aaronha01	1956	1	ML1	NL	153	153	609	106	200	34	14	26	92
9	aaronha01	1957	1	ML1	NL	151	151	615	118	198	27	6	44	132

- É também possível visualizar as colunas de uma tabela com o comando;

```
describe from batting_data
```

Query Editor My Queries Saved Queries History Tables Settings

Query Results: Unsaved Query

RESULTS

col_name	data_type	comment
playerid	string	
yearid	string	
stint	string	
teamid	string	
lgid	string	
g	string	
g_batting	string	
ab	string	
runs	int	
h	string	

Did you know? You can click on a row to select a column you want to jump to.

- Utilizando o Hive também é possível fazer junções dos dados. Digite o comando abaixo para testar uma junção;

```
select m.playerid, m.namefirst,m.namelast,
       b.yearid,b.runs
  from master_data m
 join batting_data b on (m.playerid = b.playerid )
```

Query Results: Unsaved Query

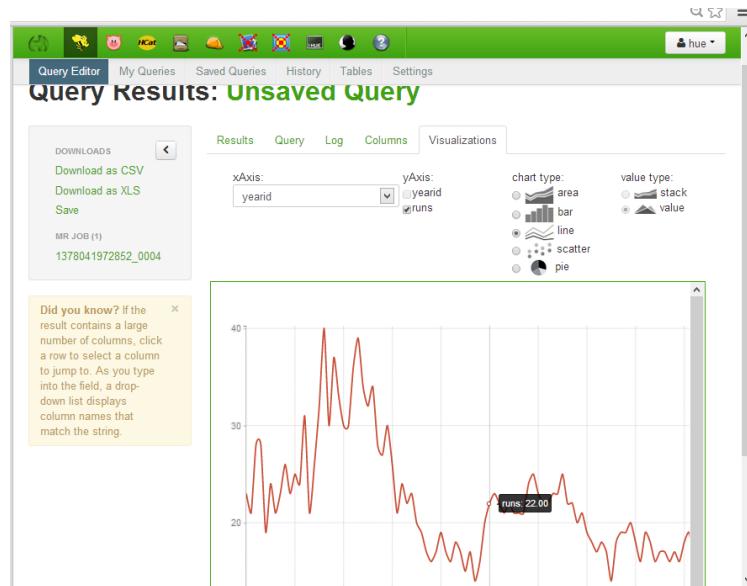
RESULTS

playerid	namefirst	namelast	yearid	runs
aaronha01	Hank	Aaron	1976	22
aaronha01	Hank	Aaron	1954	58
aaronha01	Hank	Aaron	1955	105
aaronha01	Hank	Aaron	1956	106
aaronha01	Hank	Aaron	1957	118
aaronha01	Hank	Aaron	1958	109
aaronha01	Hank	Aaron	1959	116
aaronha01	Hank	Aaron	1960	102
aaronha01	Hank	Aaron	1961	115
aaronha01	Hank	Aaron	1962	127
aaronha01	Hank	Aaron	1963	121
aaronha01	Hank	Aaron	1964	103
aaronha01	Hank	Aaron	1965	109
aaronha01	Hank	Aaron	1966	117

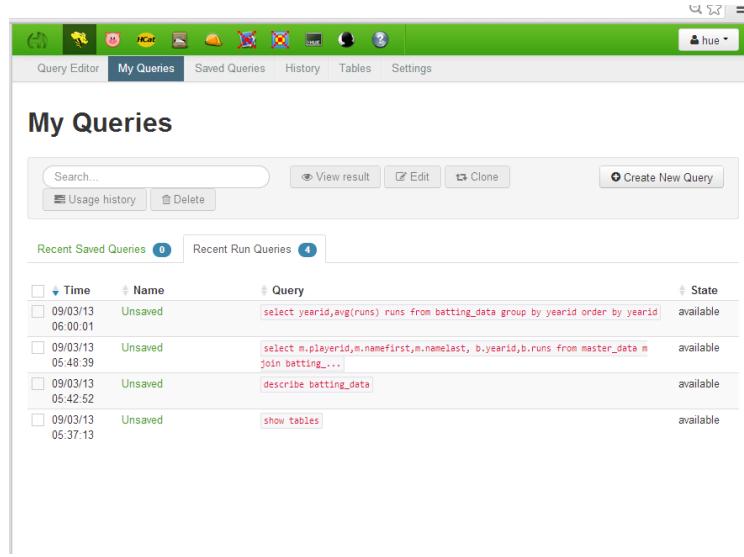
- Observe a geração dos códigos para o jobs Map Reduce. Veja o log. Observe também que é gerado um número para o processo job Map Reduce: 'MR JOB(1)';
- Outra funcionalidade é a visualização, Clique na aba 'Visualizations'. Não altere a visualização. Vamos criar um resultado mais adequado para ser apresentado:

```
select yearid,avg(runs) runs
      from batting_data
 group by yearid
 order by yearid
```

- Clique na aba 'Visualizations' e altere para um gráfico de linhas:



- Outras funcionalidades do Beeswax são armazenar ou visualizar as consultas já emitidas:



- Pode-se ainda consultar um histórico de consultas já emitidas (lembre-se que toda consulta foi convertidas para um job Map Reduce);

