NoSQL no Suporte à Análise de Grande Volume de Dados

Joel Alexandre
Licenciado Informática, Univ. Aberta, Joel.Alexandre@gmail.com

Luís Cavique
Univ. Aberta, LabMAg, Luis.Cavique@uab.pt

Resumo

Nos tempos atuais de sistemas de informação há muitos dados que são gerados a cada instante que se traduz em grandes volumes de informação. Interpretar esses dados pode dar às empresas vantagens competitivas face à concorrência. A área de *Business Intelligence* fornece às empresas mecanismos para analise desses dados. No entanto, os projetos de *software* que implementam estas soluções implicam uma grande maturidade dos requisitos. Na prática, os clientes necessitam saber que dados são importantes de serem extraídos. Alterações às análises pretendidas são habituais, no entanto implicam, muitas vezes, recursos técnicos para avaliar e evoluir a forma como os dados são tratados, aumentando a duração do projeto e, consequentemente, o respetivo custo. Este artigo vem no sentido de permitir ao utilizador final escolher uma lista de dados e dar a liberdade para organizar a informação como pretender, nomeadamente ao nível do agrupamento e ordenação de informação, isto sem a intervenção técnica para adaptar a solução às necessidades de cada utilizador.

palavras-chave: big data, NoSQL, business intelligence

Title: NoSQL in support of analysig of large volumes of data

Abstract

Current information systems have large amount of data being generated any time. A better understanding of the data can give companies competitive advantage over their rivalry. Business Intelligence offers companies mechanisms for them to analyse that data, although those software projects require maturity in the requirements specification. Changes to the requirements are usual, which have a big impact on the project cost and duration. This paper presents a solution that allows the user to specify a source of information and give him the freedom to organize the information, regards grouping and sorting, without technical skills required.

keywords: big data, NoSQL, business intelligence

1. Motivação

Nos dias actuais, grande parte dos sistemas geram informação, desde o frigorífico inteligente, passando por um sistema de acessos de transportes públicos até às plataformas de comércio electrónico com as interacções dos utilizadores ou com outros sistemas como sistemas externos de pagamento. Há cada vez mais sistemas ligados à internet, que comunicam entre si.

Aceder à informação gerada em grandes volumes, por si só, não trás grandes mais-valias às organizações. O que traz mais-valias às organizações é perceber padrões de utilização, encontrar os extremos e poder tomar decisões com base nessa informação agregada.

Esta informação serve para fazer recomendações de produtos em soluções de comércio electrónico ou detectar fraudes em pagamentos electrónicos. Pode até ser utilizada para perceber as rotas mais utilizadas e congestionadas nos transportes públicos.

Há cada vez mais fontes de dados, fontes que geram cada vez mais informação em cada instante e informação cada vez mais diversificada. Se forem aplicadas soluções tradicionais de *Business Intelligence*, as mesmas têm de ser adaptadas para cada fonte de dados. Essa situação requer bastante tempo e um conhecimento prévio da informação pretendida. Esta realidade não é de todo o pretendido pois é um processo demorado e que exige uma grande preparação.

Com o surgimento de tantas fontes de dados e a necessidade de processar toda esta informação, surgiu o conceito, o *NoSQL* [McCreary, Kelly 2013]. O *NoSQL*, ou *Not Only SQL*, permite o armazenamento, tratamento e consulta de dados não normalizados de várias formas, como será abordado mais à frente. Associado ao *NoSQL* está o conceito de *Big Data*.

2. NoSQL

O *NoSQL* [NoSQL 2013] vem trazer solução para o grande volume de informação que é gerada e que tem de ser analisada e com a necessidade das organizações manterem a informação durante um longo período de tempo, o *Big Data*.

O *Big Data* [Big Data 2013] contém as seguintes características, denominadas de 3Vs:

- Volume – com o crescimento exponencial de dados gerados, estes sistemas conseguem armazenar esse crescimento de forma sustentada.
- Velocidade – a forma como as organizações olham para os dados e o impacto que os mesmos têm no seu dia-a-dia, a informação tem de ser trabalhada num tempo reduzido, no limite, mostrar resultados em tempo real.
- Variedade – os dados armazenados podem estar nos mais diversos formatos. Desde dados em estruturas tabulares, passando por estruturas como XML ou até ficheiros binários, como um vídeo ou uma música.

Por norma, nos sistemas de *Big Data*, a entrada dos registos no sistema sofre poucas ou nenhumas alterações e só são tratados no acto de leitura/análise. Assim, não há o risco de informação que não se considera útil no acto de registo no sistema ser descartada e posteriormente se considerar que era relevante.

As soluções NoSQL [Tiwari 2011] estão divididas em alguns grupos, sendo os principais:

- Armazenamento de chave/valor (*Key*/*Value*), como Voldemort da LinkedIn ou Redis
- Armazenamento de super-colunas, como HBase, Cassandra do Facebook ou Hypertable [HBase 2013], [Cassandra 2013], [Hypertable 2013]
- Armazenamento de documentos, como XML database ou MongoDB
- Armazenamento de grafos, como HyperGraphDB ou ArangoDB
- Armazenamento orientado a objectos, como db4object

As soluções **Chave/Valor** (Key/Value) [Shinde 2013] podem-se representar como um hashmap ou array associativo. Estas têm as estruturas mais simples das soluções NoSQL e garantem a consulta da informação pela chave de forma muito eficiente, em média uma complexidade algorítmica de Θ(1). O valor não tem qualquer estrutura pré-definida, figura 1.

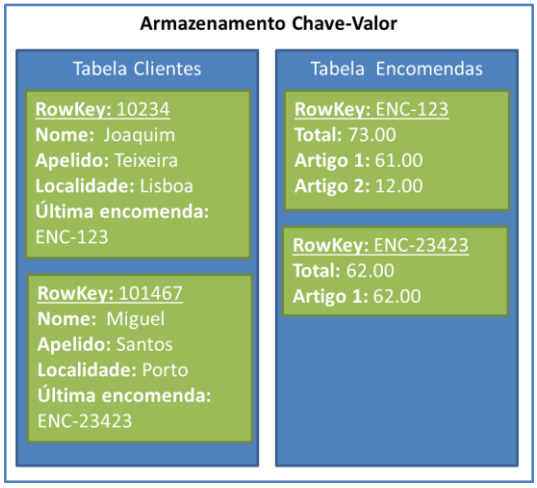

Figura 1 - Estrutura de armazenamento Chave-Valor

As soluções de **super-colunas** [Shinde 2013] permitem armazenar informação em forma de colunas, semelhantes a tabelas das bases de dados relacionais, mas sem a possibilidade de relação entre dados de linhas/tabelas diferentes. Cada linha de uma tabela pode ter estruturas distintas. Estas bases de dados permitem a pesquisa de informação por valores das colunas, figura 2.

Figura 2 - Estrutura de armazenamento super-colunas

As soluções de armazenamentos de **documentos** são semelhantes as soluções de KeyValue, mas cujo valor está num formato (por exemplo JSON – Javascript Object Notation) onde a própria solução permite a análise do valor para vários efeitos como consulta ou indexação.

Soluções de armazenamento de **grafos** que guardam relações entre dados, representados por nós e arcos, por exemplo, nas relações entre utilizadores das redes sociais. Estas soluções disponibilizam ferramentas que facilitam a consulta de relações.

As soluções de armazenamento orientado a **objectos** permitem às aplicações armazenarem objectos utilizados na programação orientada a objectos. Estes objectos podem posteriormente ser pesquisados e filtrados pelas aplicações.

Comparando soluções de chave/valor com soluções de super colunas, ambas estão preparadas para acessos optimizados pela chave, mas no caso de soluções de colunas, os valores ficam organizados por colunas, sendo as mesmas possíveis de serem filtradas pelo próprio sistema.

Ao contrário do *standard* SQL (*Structured Query Language*) para interagir com os dados dos sistemas de gestão de bases de dados relacionais que é implementado por todos os fornecedores (em níveis diferentes de conformidade) e que tem formas otimizadas para relacionar dados recorrendo a JOINs e agregar dados, para interagir com os sistemas de *NoSQL* não há uma forma única. Cada sistema tem a sua própria API ou ferramentas para permitir a consulta e manipulação de informação.

Dadas as características abordadas acima de cada sistema *NoSQL*, para o problema a resolver, a análise seria feita em sistemas de dados tabulares que permitissem a consulta a dados pelos conteúdos de cada linha, portanto, a possibilidade de filtrar pelo conteúdo

para permitir a execução de consultas *ad-hoc* e não exclusivamente por um identificador. É essencial que permita a execução de processos assíncronos de grande volume de informação, conhecidos como processos de *MapReduce*.

Neste sentido, a análise é limitada às soluções Hadoop/HBase, Cassandra e Hypertable. Dado que tanto as soluções Cassandra e Hypertable assentam sobre Hadoop para suportar operações de *MapReduce*, foi decido utilizar um ecossistema único, Hadoop/Hbase.

MapReduce

O MapReduce [Dean, Ghemawat 2004]é um conjunto de processos que permite tratar um grande volume de informação de forma distribuída. O conceito prende-se com duas operações: *Map* e *Reduce*, figura 3. A função do operador *Map* é dividir o problema em subconjuntos mais pequenos que possam ser distribuídos por outros nós do *cluster*. A operação *Reduce* pega em todos os subconjuntos de dados, agrega e trata a informação e responde ao problema original. Ao contrário do SQL, que armazena os resultados em memória e os retorna ao cliente sem os armazenar de forma permanente, os processos de MapReduce armazenam os dados de forma persistente para posteriormente serem consultados.

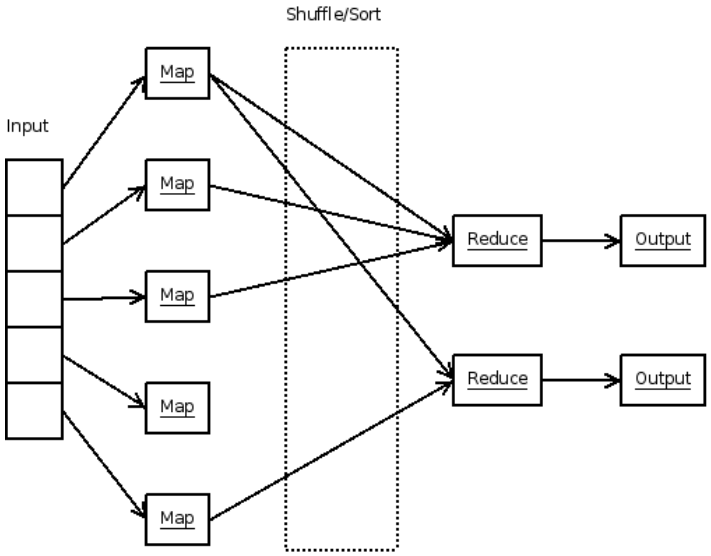

Figura 3 - Fluxo do processo MapReduce

Hadoop

O Hadoop [Hadoop 2013] é um ecossistema que permite a gestão, armazenamento e tratamento de ficheiros de grande volume e de forma distribuída e redundante. O Hadoop tem os seguintes componentes funcionais – o *Job Traker*, um processo central que gere e coordena o processamento paralelo dos diferentes nós; o *Task Tracker*, um processo que faz o processamento de informação em cada nó; o *Name Node*, um processo central que gere e coordena o armazenamento da informação; o *Data Node*, que trata do armazenamento de informação em cada nó.

HBase

O sistema HBase [Apache SF 2013], [George 2011] é um sistema de armazenamento NoSQL do tipo super-coluna assente sobre Hadoop. Isto é, utiliza o Hadoop para armazenar os dados. Também é possível utilizar o MapReduce do Hadoop para processar dados.

Os termos/componentes utilizados no HBase são os seguintes:

- Tabela / *Table* – O HBase organiza a informação em tabelas. Os nomes das tabelas são Strings.
- Linha / *Row* – Dentro de cada tabela os dados são armazenados de acordo com a sua linha. Cada linha é identificada univocamente pela sua *rowkey*, chave da linha. As chaves das linhas não têm tipo e são tratados como um *array* de *bytes*.
- Família de coluna / *Column Family* – Os dados dentro de cada linha estão agrupados por família de coluna. Estas famílias têm de ser definidas aquando da criação das tabelas e não são facilmente modificados. Cada linha de uma tabela tem as mesmas famílias de coluna, no entanto podem não armazenar qualquer informação. Cada família de colunas fica armazenada em ficheiros distintos. Os nomes da família de colunas é uma String.
- Coluna / *Column Qualifier* – Os dados dentro de cada família de coluna são armazenados e acedidos através da sua coluna. As colunas não precisam de ser definidas aquando da criação da tabela e cada linha pode ter colunas distintas. Os dados armazenados não têm tipo e são tratados como um *array* de *bytes*.
- Célula / *Cell* – Cada combinação de uma *rowkey*, família de coluna e coluna identifica univocamente uma célula. Os valores de uma célula também não têm tipos e são tratados como um *array* de *bytes*.
- Versão / *Version* – Os valores dentro das células são versionados pelo *timestamp* (um long). Quando o *timestamp* não é especificado, o *timestamp* de quando a operação é realizada é utilizado. Quando os valores são consultados não é necessário especificar a versão, sendo que nesses casos, a versão mais recente é a retornada.

Podemos considerar que uma tabela em HBase é uma matriz de 4 dimensões. Um exemplo de um registo de vendas seria representado na tabela 1.

Tabela 1 - Organização de uma tabela HBase

	Família de colunas – info			
Rowkey	**Data**	**Loja**	**Distrito**	**Volume Vendas**
Linha A	01/05/2013	Oeiras	Lisboa	5000.00
Linha B	01/05/2013	Cascais	Lisboa	5900.00
Linha C	02/06/2013	Cascais	Lisboa	TS:1372010537=6100.00 TS:1372010530=6150.00
Linha D	02/06/2013	Lamego	Viseu	4500.00

Cada célula contém pelo menos uma versão dos dados, mas pode conter mais versões identificadas pelo *timestamp*

No HBase apenas a *rowkey* está indexada. Não é possível o HBase criar índices secundários, sendo no entanto possível criá-los aplicacionalmente. A *rowkey* está ordenada lexicograficamente.

OLAP

Mesmo recorrendo a conceitos de NoSQL, faz todo o sentido utilizar alguns conceitos de Cubos OLAP, nomeadamente, dimensões e medidas. Durante a fase de importação dos dados, o cubo é definido e, em particular, são ainda definidas quais as dimensões e medidas existentes, sendo que todas as colunas, numéricas ou não, podem ser dimensão ou medida, cabendo ao utilizador indicar como devem ser consideradas na análise. Após essa definição cabe ao utilizador final organizar e consultar a informação com base nas dimensões e escolher que medida pretende visualizar. As medidas contabilizam as seguintes possibilidades:

- count – contagem do número de elementos
- sum – soma do valor dos elementos
- max – o valor máximo dos elementos
- min – o valor mínimo dos elementos
- avg – a média dos valores dos elementos

3. Caso de Estudo

Sendo o objectivo a análise de dados não normalizados/estruturados, o processo é definido na figura 4.

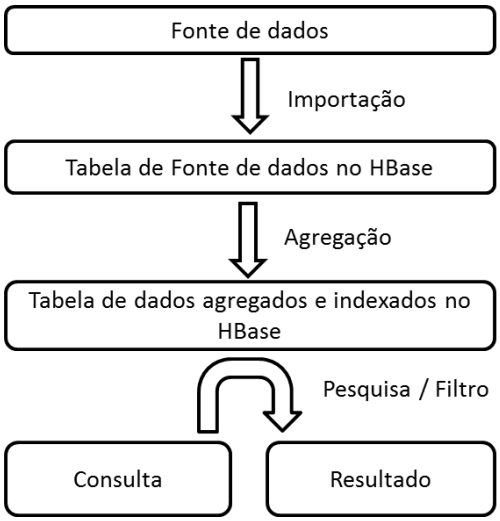

Figura 4 - Processo desde a fonte de dados até ao resultado da agregação

Os dados são importados a partir da sua fonte para uma tabela HBase. Posteriormente, de acordo com as necessidades de agregação, há um processo de MapReduce que efectua a agregação e guarda os resultados agregados e indexados numa tabela própria onde podem ser efectuadas as pesquisas sobre esses dados.

Esta solução vai conter dois tipos de tabelas – tabelas que contêm os dados importados das fontes de informação e tabelas de agregação, que armazenam os dados de agregação que vão sendo calculados. As tabelas com os dados importados têm o nome que é

definido pelo utilizador aquando da sua importação. As tabelas de agregação têm o mesmo nome que a tabela da fonte, concatenado com o seguinte texto no final "-AGG".

Visto que aquando da criação das tabelas é necessário definir o nome das mesmas e a família das colunas mas como nesta solução não faz sentido separar as colunas por diversas famílias, é definida uma única família com o nome "cf". No caso das tabelas de agregação, onde os resultados agregados ficam armazenados, a família definida é "AGG".

Agregação

De seguida, é apresentada a forma de agregar os dados e como são armazenados para consulta. As fontes de dados estão armazenadas em tabelas com a estrutura referida na tabela 2.

Tabela 2 - Tabela HBase

Rowkey	**Coluna A**	**Coluna B**	**…**	**Coluna N**
Linha 1	Valor A 1	Valor B 1		Valor N 1
Linha 2	Valor A 2	Valor B 2		Valor N 2
…				
Linha M	Valor A M	Valor B M		Valor N M

Para fazer a agregação tem de se especificar quais as colunas que devem ser tratadas como as dimensões e quais tratadas como medidas. As linhas são agrupadas pelos valores das colunas consideradas como dimensões. Este é a tarefa do *Map*, que junta todas as combinações distintas dos valores das dimensões.

Após esta fase, as medidas são calculadas para cada um dos grupos. Este é a fase de Reduce. São calculados, para cada uma das medidas, o total, o número de linhas, a média, o máximo e o mínimo. Os dados, depois de calculados, são armazenados de forma persistente para poderem ser consultados, sem ter de haver nova agregação. Como referido anteriormente, só as *rowkeys* das tabelas do HBase estão indexadas, estando ordenadas de forma lexicográfica.

Por forma a possibilitar a consulta, não só dos dados agregados na sua totalidade, mas também as extremidades das métricas, isto é, quais as melhores ou piores métricas, é preciso criar uma estrutura específica para as *rowkeys* para que seja possível obter esses dados a partir da agregação já calculada de forma eficiente.

É preciso responder a questões como "para as dimensões A e B, quais as 10 melhores médias de C?" ou "para as dimensões A e B, qual a pior total de D?". A API do HBase permite obter, de forma eficiente, as linhas a partir de uma determinada *rowkey* até outra *rowkey* ou até atingir um número especificado de linhas. Assim, a *rowkey* tem de conter a informação necessária para identificar as dimensões, métrica e informação relevante para a ordenação.

Como a ordenação é lexicográfica, a informação contida na *rowkey* tem de conter, da esquerda para a direita, a informação mais genérica identificadora da linha agregada, até aos dados mais específicos. Assim, a seguinte estrutura de *rowkey* permite a consulta de dados agregados consoante a métrica, para cada métrica e ordenação (ascendente ou descendente):

<Nome de dimensão 1>(…)<Nome de dimensão N> <Nome métrica agregada> <Tipo agregação da métrica (total, contagem, máximo, mínimo ou média)> <Ordenação do valor da métrica (ascendente ou descente)> <Valor da métrica> <Valor da dimensão 1>(…)<Valor da dimensão N>

Na tabela 3, cada linha contém uma coluna para a respectiva métrica com todas as agregações possíveis.

Tabela 3- Estrutura dos dados agregados

cf:<Nome da métrica agregada>

{
"count": <valor>,
"max": <valor>,
"min": <valor>,
"sum": <valor>,
"avg": <valor>,
"metric": <nome da métrica agregada>
}

Para fazer uma pesquisa de agregações, é preciso indicar as dimensões, a métrica que pretendemos, o tipo de agregação a ordenar e a direcção da ordenação. Na tabela 4, os restantes dados não são especificados na pesquisa, pelo que assim obtemos os resultados ordenados.

Tabela 4 - Formato da pesquisa de resultados agregados

<Nome de dimensão 1>(…)<Nome de dimensão N> <Nome métrica agregada> <Tipo agregação da métrica (total, contagem, máximo, mínimo ou média)> <Ordenação do valor da métrica (ascendente ou descente)>

O <Valor da métrica>, especificado na *rowkey* da agregação, é o valor pelo qual os resultados devem ser ordenados. Neste caso, se a ordenação for ascendente, este valor é o valor da agregação. Mas se a ordenação for decrescente, quer isto dizer que os valores maiores têm de aparecer em primeiro lugar, este valor é resultado de um número muito grande (aqui foi considerado este número como 999 999 999 999 999) subtraído do valor da agregação. Passa assim a ser possível obter a ordenação descendente, mas mantendo a característica da ordenação lexicográfica das *rowkeys*.

Nota: em vez de 999 999 999 999 999, poderia usar alguma constante como Double. MAX_VALUE ($\approx 8x10^{307}$) ou Long.MAX_VALUE ($=2^{63}-1$), mas eram valores demasiado elevados, isto é, com muitos dígitos, para serem usados de forma prática.

De forma a poder garantir que, independentemente da ordem de introdução das dimensões, não há duplicação de agregações, a referência das dimensões na *rowkey* é ordenada alfabeticamente.

Para demonstrar, foram utilizadas as informações sobre todos os voos comerciais com origem ou destino nos EUA entre 2000 e 2005, disponíveis "On-Time Performance" disponíveis em http://www.transtats.bts.gov/[2]. Cada registo pode conter até 109 indicadores sobre cada voo. Ao todo, o período indicado, contempla próximo de 30 milhões de registos.

Após a importação dos dados dos ficheiros originais para uma tabela HBase, passou a ser possível fazer agregação. Dos diversos indicadores de cada voo, podemos analisar o ano do voo e pela transportadora e analisar o tempo de atraso da partida face ao previsto. Considerando as colunas *Year* e *UniqueCarrier* como dimensões e a coluna *DepDelayMinutes* como métrica e correndo o processo de MapReduce da agregação, é possível ver os seguintes resultados, ordenados pela média de atraso ordenando de forma descendente na tabela 5.

Tabela 5 - Resultado da consulta ao resultado de agregação

```
scan <tabela de agregações>,  {LIMIT => 2, STARTROW=>
'Year UniqueCarrier DepDelayMinutes avg desc'}
```

Rowkey	Valor

Year
UniqueCarrier DepDelayMinutes
avg
desc
999999999999981,3950
2000
UA
	{
	"count":776559,
	"max":1277.0,
	"min":0.0,
	"sum":1.3671339E7,
	"avg":17.605022928071143
	}

Year
UniqueCarrier
DepDelayMinutes
avg
desc
999999999999984,39082
2000
HP
	{
	"count":219160,
	"max":729.0,
	"min":0.0,
	"sum":3201748.0,
	"avg":14.60918050739186
	}

Podemos verificar que, no ano 2000, a companhia UA teve o pior registo de atrasos na partida, com uma média de 17,6 minutos de atraso. Para esse mesmo registo, podemos verificar que, nesse ano, o pior atraso dessa companhia foi de 1277 minutos. Podemos ver qual a companhia com o voo mais curto ao considerar como dimensão a companhia aérea e como métrica a duração do voo (*AirTime*) e executar o respetivo processo de MapReduce.

A consulta é:

```
scan <tabela de agregações>,  {LIMIT => 1, STARTROW=>
'UniqueCarrier AirTime max asc'}
```

[1] http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time
[2] http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time

4. Conclusões

Este artigo tem como objetivo implementar uma ferramenta que melhore os processos de consulta de dados não normalizados de grande volume. Os projetos de análise de dados tendem a ser dispendiosos, quer em tempo, quer em custos, pois não é fácil o cliente perceber o que realmente pretende sem antes ver alguns dos dados. Mesmo após a decisão do cliente, à medida que o cliente percebe a utilidade dos resultados, pretende fazer alterações. Também as bases de dados de suporte a cubos não estão preparadas para grandes volumes de informação, ou pelo menos não escalam horizontalmente com facilidade.

Para dar resposta à solução, e após analisar as diversas tecnologias de *Big Data* e *NoSQL*, a escolha de Hadoop e HBase pareceu bastante óbvia, quer por permitir dados não estruturados, quer por suportar dados colunares, isto é, com colunas e por escalar de forma horizontal. Foi possível também verificar que o Hadoop e o HBase são bastante utilizados por empresas de renome para tratar e analisar grandes volumes de informação.

É importante salientar que as tecnologias de Hadoop e HBase estão em contante evolução. Por exemplo, durante a execução da solução passou a estar disponível na versão estável os co-processadores no HBase, que se podem traduzir como *triggers* das base de dados relacionais. Isto poderia permitir actualizações incrementais dos dados com resultados em tempo real.

A solução apresentada está dividida em duas áreas: importação de dados e a agregação dos dados. Para a **importação dos dados**, implementou-se um processo simples, que não obriga a configurações por parte do utilizador. Possibilita a importação de ficheiros de texto em dois formatos, CSV e multilinha. Este suporta um número bastante elevado de colunas (a limitação será do sistema e não do Hadoop/HBase). Para **agregação dos dados** foi utilizado o motor do Map/Reduce do Hadoop/HBase, um conceito introduzido pela empresa Google para o tratamento de grandes volumes de informação, sem necessidade de ter uma estrutura rígida.

Este motor permite tratamentos assíncronos de informação contida em tabelas HBase ou ficheiros Hadoop, ao separar o tratamento de dados em duas componentes: Map e Reduce. Devido às características do HBase, as tabelas não suportam índices ao nível da informação contida nas colunas de forma nativa, tendo apenas o identificador da linha (*rowkey*) indexado. Assim, o ponto fulcral desta solução foi identificar a forma de poder filtrar os resultados da agregação rapidamente, independentemente da dimensão das fontes dos dados. Para poder tirar partido do HBase, os identificadores dos resultados agregados devem ser construídos a partir dos nomes das colunas, as ordenações possíveis e os valores das células agregadas. Com esta estrutura é possível consultar a informação de forma rápida e organizada.

Em conclusão, os sistemas de *NoSQL*, com as suas diferentes abordagens, conseguem dar respostas às necessidades de informação actuais. Isto sem haver grande preocupação em tratar os dados quando são gerados, mas apenas quando são consultados, evitando descartar informação.

Referências

Apache Software Foundation, The Apache HBase Reference Guide, http://hbase.apache.org/book/rowkey.design.html (acedido dezembro de 2013)

Big Data, http://www.gartner.com/it-glossary/big-data/ (acedido dezembro de 2013)

Cassandra, https://cassandra.apache.org (acedido dezembro de 2013)

Dean J., S. Ghemawat, 2004, MapReduce: Simplified Data Processing on Large Clusters, OSDI'04 Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation, volume 6, pp. 10-10.

George L., 2011, HBase: The Definitive Guide, Random Access to your planet-size data, O'Reilly Media.

Hadoop, http://hadoop.apache.org (acedido dezembro de 2013)

HBase, http://hbase.apache.org (acedido dezembro de 2013)

Hypertable, http://hypertable.org (acedido dezembro de 2013)

McCreary D., A. Kelly, 2013, Making Sense of NoSQL: A guide for managers and the rest of us, Manning Publications.

NoSQL, http://www.techopedia.com/definition/27689/nosql-database (acedido dezembro de 2013)

Shinde S., SQL Server Business Intelligence and Big Data - What is Wide Column Stores? 2013. http://bi-bigdata.com/2013/01/13/what-is-wide-column-stores/ (acedido em dezembro de 2013)

Shinde S., SQL Server Business Intelligence and Big Data, What is Key-Value Stores? 2012. http://bi-bigdata.com/2012/12/25/what-is-key-value-stores/ (acedido em dezembro de 2013)

Tiwari S., 2011, Professional NoSQL, John Wiley & Sons, Inc.

Joel Alexandre, Licenciado pela Universidade Aberta em Informática, foi programador e administrador de sistemas na Construlink nas suas soluções de B2B. Posteriormente integrou a Safira onde desenvolveu soluções na área dos seguros na Caixa Seguros. Foi também Chief Technical Officer na Gatewit onde geriu a equipa e as soluções tecnológicas dos produtos B2G e B2B da empresa. Actualmente é Consultor Sénior na Affinity, estando integrado nas soluções de pagamento electrónico da PT Comunicações no Sapo.

Luís Cavique, Professor Auxiliar no Departamento de Ciências e Tecnologia (DCeT), Secção de Informática, Física e Tecnologia (SIFT). Coordenador da Licenciatura em Informática no biénio 2012 - 2014. Licenciado em Engenharia Informática em 1988 pela FCT-UNL. Obteve o grau Mestre em Investigação Operacional e Eng. Sistemas pelo IST-UTL em 1994. Obteve o grau de Doutor em Eng. Sistemas pelo IST-UTL em 2002. Tem como áreas de interesse, a intersecção da Informática (Computer Science) com a Engenharia de Sistemas (Management Science) designadamente a área de "Data and Graph Mining".