Social Media Analytics App

André Dinis[1], Erbi Silva[1], Ricardo Simões[1]

[1] Departamento de Engenharia Informática,
Faculdade de Ciências e Tecnologia,
Universidade de Coimbra

Abstract. Este é um projecto que assenta no tema *Social Media Analytics App* presente no livro *Cloud Computing: A Hands-On Approach* com o acrescento dos serviços Glacier, CloudSearch, SES e DynamoDB. Existe uma aplicação que recolheos tweets com contenham uma determinada keyword e coloca-os numa queue SQS. Existe uma outra aplicação que consome a informação da SQS e envia-a para a DynamoDB, CloudSearch e Glacier. Existe ainda uma terceira aplicação que faz procuras de determinada keyword na CloudSearch para obter as informações que se encontram guardas sobre a mesma. Encontram-se apresentado no artigo um esquema explicativo do funcionamento so sistema. Apresentamos também algumas análises bem como aspetos que achamos que seriam uma mais-valia futura para a nossa aplicação.
Palavras-chave: *Amazon, Amazon Web Services, AWS, DynamoDB, CloudSearch, Glacier.*

1 Introdução

Este projeto tem como ponto de partida o tema *Social Media Analytics App*, presente no livro *Cloud Computing: A Hands-On Approach* [1]. Para a sua realização são utilizados alguns dos serviços da *Amazon Web Services* (AWS) [2]. É espectável que os membros do grupo adquiram e aprofundem conhecimentos acerca dos referidos serviços, sendo esse um dos principais objetivos do projeto.
A ideia base do projeto passa por fazer uma análise estatística dos *tweets* publicados no Twitter, através da pesquisa por *keywords*, assim como o sentimento associado aos *tweets*.
Durante o resto do documento, é apresentada uma breve descrição do problema e a abordagem seguida para a sua resolução.

2 Problema

2.1 Descrição do Problema

Existe atualmente uma grande concorrência entre empresas dos mais variados ramos, sendo por isso importante conhecer e saber o que dizem os clientes e potenciais clientes de uma determinada empresa e das suas empresas concorrentes.

Posto isto, imagine-se o cenário em que determinada empresa pretende avaliar a sua projeção no mercado, isto é, nos seus clientes e potenciais clientes. Imagine-se ainda um cenário semelhante, em que a referida empresa lança um produto novo ou faz algum tipo de promoção e pretende avaliar o impacto que isso tem no mercado. Apesar da importância que este tipo de análise pode ter, não existe propriamente uma forma simples de satisfazer estas necessidades das empresas.

Parte dessas necessidades pode ser alcançada através de uma análise a estatísticas de uma rede social. Neste projeto em concreto, é feita uma pesquisa por uma ou várias *keywords* no Twitter, elaborando um conjunto de estatísticas relacionadas com *tweets* que contêm a(s) *keyword*(*s*), havendo também uma análise da correspondência entre o *tweet* e o sentimento presente nele. Deste modo, as empresas podem, por exemplo, verificar se estão a falar delas, o que dizem e se provocam bons ou maus sentimentos nos que falam sobre elas. Podem analisar também o impacto que o lançamento de um novo produto ou promoção teve nos *stakeholders*.

2.2 Desafios a superar

Um dos grandes problemas a resolver prende-se com a escolha correta de serviços que encaixem no projeto e encontrar um sentido para a sua utilização. A Amazon oferece um vasto leque de serviços mas são muitos aqueles que não se adequam de uma forma tão correta, para que façam sentido no nosso projeto.

3 Abordagem

3.1 Arquitetura

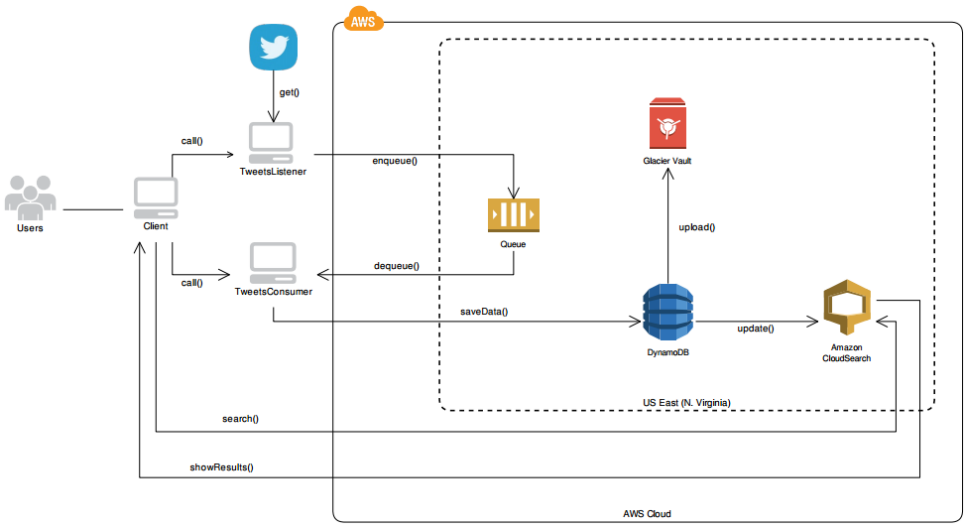


Figura 1 - Arquitetura da *Social Media Analytics App*

A *Social Media Analytics App* é composta por vários componentes, visíveis na figura em cima (Fig. 1).
Os utilizadores acedem a uma aplicação cliente (*Client*) que comunica com duas outras aplicações - *TweetsListener* e *TweetsConsumer* - e com um serviço da *Amazon Web Services* (*Amazon CloudSearch*).
A aplicação *TweetsListener* comunica com a API do Twitter e com um serviço de filas da AWS (*Simple Queue Service*).
A aplicação *TweetsConsumer* comunica com a fila ainda agora referida e com o serviço de base de dados não relacional da AWS (*DynamoDB*).
Há ainda dois serviço da AWS (*CloudSearch* e *Glacier*) que interagem com o serviço *DynamoDB*.
Na subsecção 3.2 são apresentados com mais detalhe os serviços da AWS usados, e na subsecção 3.3 é apresentado o modo de funcionamento da aplicação e como tudo o que é apresentado na arquitetura se interliga.

3.2 Uso de Serviços AWS

É apresentada de seguida uma breve descrição dos serviços da AWS usados, tanto os lecionados nas aulas como os serviços extra.

Simple Queue Service (SQS) [3] - O *Amazon Simple Queue Service* (SQS) é um serviço de fila de mensagens rápido, confiável e escalável. Pode ser usado para transmitir qualquer volume de dados, havendo no entanto um limite no tamanho das mensagens (256 KB). Isto pode ser feito sem perda de mensagens e sem exigir que outros serviços estejam sempre disponíveis.
O SQS permite que quem o utilize não tenha de levar com a sobrecarregar administrativa de operar e dimensionar um cluster de sistema de mensagens altamente disponível, sendo pago um preço baixo apenas pelo que é efetivamente utilizado.
No projeto em questão, este serviço foi utilizado para transmitir mensagens entre o *Listener* e o *Consumer*. As mensagens transmitidas são os *tweets* recolhidos pelo *Listener*, em formato JSON.

DynamoDB [4] - O *Amazon DynamoDB* é um serviço de base de dados NoSQL rápido e flexível, que fornece escalabilidade facilmente. Foi projetado para oferecer desempenho consistente e rápido. Normalmente, a latência média do serviço é inferior a 10 milissegundos. Oferece suporte a estruturas de dados de documentos e chave-valor, proporcionando a flexibilidade para criar a arquitetura ideal para determinada aplicação.
Dependendo do tipo de utilizador, o *DynamoDB* pode ter diferentes tipos de utilização. No caso de um *developer*, o *DynamoDB* pode ser usado para criar uma tabela que pode guardar e devolver qualquer quantidade de dados, e servir qualquer número de pedidos. O *DynamoDB* propaga automaticamente os dados e o tráfego de acesso à tabela a um número suficiente de servidores para lidar com a capacidade de pedidos solicitada pelo cliente e a quantidade de dados armazenados, mantendo um desempenho consistente e rápido. Todos os dados são guardados em SSDs (*Solid State Disks*) e são automaticamente replicados por múltiplas "*Availability Zones*" numa região para fornecer alta disponibilidade e durabilidade dos dados.
No caso de um administrador de base de dados, o *DynamoDB* pode ser usado para criar uma nova tabela, fazer "*scale up*" ou "*scale down*" à capacidade requisitada para a tabela sem ter *downtime* ou degradação de performance.
No projeto em questão, o *DynamoDB* foi utilizado para guardar os dados relativos às estatísticas do Twitter.

CloudSearch [5] - O *Amazon CloudSearch* é um serviço de procura na *cloud* que permite que os utilizadores integrem facilmente uma funcionalidade de procura rápida e altamente escalável nas suas aplicações.
Com o *Amazon CloudSearch* não é necessário haver preocupação com fornecimento, configuração e manutenção de hardware. É possível criar um domínio de procura e fazer *upload* dos dados em que queremos efetuar a procura de forma simples e rápida,

e o *Amazon CloudSearch* irá disponibilizar automaticamente os recursos necessários e implementar um índice de pesquisa altamente sintonizado.
Pode-se mudar facilmente os parâmetros de pesquisa, ordenar os resultados, etc. À medida que o volume de dados e tráfico varia, o *Amazon CloudSearch* escala similarmente para satisfazer as necessidades do utilizador.

No projeto em questão, o *CloudSearch* foi usada para pesquisar os dados relativos às estatísticas do Twitter que estão no *DynamoDB*. Sempre que se adicionam novos dados ao *DynamoDB*, são também atualizados os dados de procura do *CloudSearch*.

Glacier [6] - O *Amazon Glacier* é um serviço seguro, durável e extremamente de baixo custo para arquivo de dados e *backup online*. Os clientes deste serviço podem-no usar para guardar pequenas ou grandes quantidades de dados por tão pouco quanto 0.009€ por gigabyte por mês, um custo muito menor quando comparado com outras soluções. Para manter os custos baixos, o *Amazon Glacier* está otimizado para acessos a dados não muito frequentes, pelo que o tempo que os ficheiros demoram a ficar disponíveis (tipicamente 4 a 6 horas) é aceitável.
No projeto em questão, o *Glacier* é usado para fornecer a funcionalidade de *backup* à base de dados. Sendo um serviço auxiliar de *backup* à base de dados, onde não é necessário aceder imediatamente aos dados, é uma solução bastante interessante e que satisfaz as nossas necessidades.
No entanto, não conseguimos acabar na totalidade o que tínhamos idealizado para este serviço, uma vez que tínhamos em mente o uso de mais um serviço da AWS - *Email Sending Service* (SES) - que iria trabalhar em conjunto com o *Glacier*. Os detalhes desta nossa ideia podem ser encontrados na secção 5.1 (Melhorias/Trabalho Futuro).

3.3 Funcionamento da Aplicação

Os utilizadores acedem à aplicação cliente (*Client*) e têm 2 opções:

1. Fazer download de um novo conjunto de tweets;
2. Fazer uma pesquisa nos dados que já existem.

Em ambas as opções, os utilizadores terão que indicar a *keyword* pela qual querem pesquisar.
Se for escolhida a primeira opção, a aplicação cliente invoca duas outras aplicações: *TweetsListener* e *TweetsConsumer*.
A aplicação *TweetsListener* usa a API do Twitter para obter *tweets* com base na *keyword* definida pelo utilizador e coloca-os numa fila através do serviço *Simple Queue Service* (SQS) da AWS.
Por sua vez, a aplicação *TweetsConsumer* acede à mesma fila para obter os *tweets*, analisa-os e guarda os resultados agregados numa base de dados não relacional da AWS (*DynamoDB*). Depois dos dados serem adicionados à base de dados, o serviço *CloudSearch* atualiza os seus documentos de procura através do *DynamoDB*. Além disso, é criado um ficheiro com os mesmos dados que foram adicionados à base de dados e é feito o seu *upload* para o *Glacier*.
Se for escolhida a segunda opção, a aplicação cliente acede ao *CloudSearch* e pesquisa pela *keyword* definida pelo utilizador.

3.4 Excertos de Código

Consideramos importante a inclusão de 2 excertos de código no presente relatório, relativos à inserção de dados na base de dados não relacional (*DynamoDB*) e à atualização dos documentos de procura do serviço *CloudSearch*. Estas 2 funções são corridas em sequência, isto é, uma a seguir à outra, pela ordem aqui apresentada.
Assim, apresentamos de seguida os referidos excertos, assim como uma breve descrição de cada um.

3.4.1 DynamoDB

```
# This function adds an item to DynamoDB
def addItem(valuedic):
    print 'Inserting data on DynamoDB...'
    # Define statistics variables
    id_stat = valuedic['_id']
    total_tweets = valuedic['totaltweets']
    positive_tweets = valuedic['positivesentiment']
    neutral_tweets = valuedic['neutralsentiment']
    negative_tweets = valuedic['negativesentiment']
    hashtags = valuedic['hashtags']
    top_tweets = valuedic['toptweets']
    total_retweets = valuedic['totalretweets']
    metadata = valuedic['metadata']
    hourly_aggregate = valuedic['hourlyaggregate']
    table = connDB.get_table('twitter_stats_table')
    item_data = {
        'total_tweets': str(total_tweets),
        'positive_sentiment': str(positive_tweets),
        'neutral_sentiment': str(neutral_tweets),
        'negative_sentiment': str(negative_tweets),
        'hashtags': str(hashtags),
        'top_tweets': str(top_tweets),
        'total_retweets': str(total_retweets),
        'hourly_aggregate': str(hourly_aggregate)
    }


    global hashDB
    global rangeDB
    hashDB = str(id_stat)
    rangeDB = str(metadata)

    item = table.new_item(
        hash_key=hashDB,
        range_key=rangeDB,
        attrs=item_data
    )
    item.put()
    print 'Data was successfully inserted!'
```

Figura 2 - Excerto de código do DynamoDB

Esta função serve para adicionar um item à base de dados e é chamada na aplicação *TweetsConsumer*, depois de todas as estatísticas serem criadas e guardadas num

dicionário (*valuedic*, neste caso). Começam-se por definir todas as variáveis relativas a estatísticas através do referido dicionário, são definidas as chaves para inserção na base de dados (*hashDB* e *rangeDB*), é criado o item a inserir com recurso às chaves ainda agora referidas e aos dados com as estatísticas e, finalmente, o item é inserido na base de dados.

3.4.2 CloudSearch

```
# This function updates the search documents of CloudSearch
def update_cloudsearch():
    # Get the result from table
    results = table.get_item(hash_key=hashDB, range_key=rangeDB)

    # Connect to CloudSearch
    print "Connecting to CloudSearch..."
    connCS = boto.connect_cloudsearch2(aws_access_key_id='AKIAIXQMLV2XPFQ2NFRQ',
                                       aws_secret_access_key='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')
    print "Connected!\n"

    # Search domain
    print "Searching for domain..."
    domain = connCS.lookup('twitter-app')
    print "Domain founded: ", domain.name

    print "Updating CloudSearch..."
    doc_service = domain.get_document_service()
    doc_service.add(results.get('id_stat'), results)
    doc_service.commit()
    print "Updated!"
```

Figura 3 - Excerto de código do CloudSearch

Esta função serve para atualizar os documentos de pesquisa do *CloudSearch*, com base no item que acabou de ser inserido na base de dados.

Inicialmente obtém-se o referido item da tabela da base de dados, através das chaves definidas anteriormente (*hash_key* e *range_key*), ficando este guardado na variável *results*.

Depois é feita uma conexão ao *CloudSearch,* ficando essa conexão associada à variável *connCS*.

Posteriormente procura-se pelo domínio de procura do *CloudSearch*, através da função *lookup()*. O domínio fica guardado na variável *domain*.

Finalmente é feito o *update* aos documentos de pesquisa do *CloudSearch*.

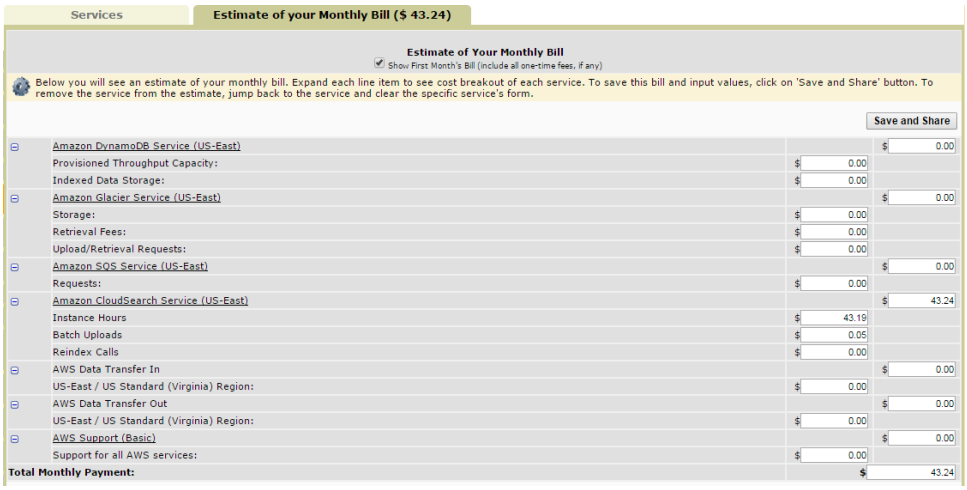
4 Análise

4.1 Análise da Confiabilidade

Não há muito que se possamos afirmar em relação à confiabilidade do programa. No entanto, em termos gerais, pode-se dizer que o programa é confiável, uma vez que tolera inputs errados ou inesperados, nomeadamente tipos errados de variáveis ou fora dos limites apresentados.

4.2 Análise do Custo

Para o cálculo do custo, consideramos que iriam haver 3 pedidos por dia, com 5 palavras diferentes, totalizando assim 15 pedidos por dia (450 pedidos por mês).
O valor resultante do cálculo pode ser verificado na seguinte figura:



Services | Estimate of your Monthly Bill ($ 43.24)

Estimate of Your Monthly Bill

Show First Month's Bill (include all one-time fees, if any)

Below you will see an estimate of your monthly bill. Expand each line item to see cost breakout of each service. To save this bill and input values, click on 'Save and Share' button. To remove the service from the estimate, jump back to the service and clear the specific service's form.

Save and Share

Amazon DynamoDB Service (US-East)		$ 0.00
Provisioned Throughput Capacity:	$ 0.00	
Indexed Data Storage:	$ 0.00	
Amazon Glacier Service (US-East)		$ 0.00
Storage:	$ 0.00	
Retrieval Fees:	$ 0.00	
Upload/Retrieval Requests:	$ 0.00	
Amazon SQS Service (US-East)		$ 0.00
Requests:	$ 0.00	
Amazon CloudSearch Service (US-East)		$ 43.24
Instance Hours	$ 43.19	
Batch Uploads	$ 0.05	
Reindex Calls	$ 0.00	
AWS Data Transfer In		$ 0.00
US-East / US Standard (Virginia) Region:	$ 0.00	
AWS Data Transfer Out		$ 0.00
US-East / US Standard (Virginia) Region:	$ 0.00	
AWS Support (Basic)		$ 0.00
Support for all AWS services:	$ 0.00	
Total Monthly Payment:		$ 43.24

Figura 4 - Análise de Custo

5 Discussão

5.1 Melhorias/Trabalho Futuro

No futuro seria interessante efetuar algumas alterações ao programa, de forma a que lhe fosse adicionada uma mais valia.
Uma possibilidade seria a integração do SES (*Email Sending Service*) com o *Glacier*, de forma a que quando um ficheiro estivesse disponível, isto é, quando um ficheiro acabasse o *upload* para o *Glacier*, o utilizador receberia um email com a informação de como aceder ao ficheiro ou fazer o seu *download*.
Para uma melhor performance e para manter a informação atualizada, as aplicações *TweetsListener* e *TweetsConsumer* poderiam estar a correr no EC2. Também a aplicação cliente (*Client*) poderia ser colocada a correr neste serviço, passando a ser um cliente *web* com uma interface gráfica.
Além disso, o nosso ideal seria que a aplicação cliente pudesse correr automaticamente, recolhendo assim os dados do Twitter sem intervenção humana. A única coisa que os utilizadores da nossa aplicação teriam de fazer, seria aceder a ela para fazer pesquisas aos dados já recolhidos.

6 Conclusão

Os serviços da *Amazon Web Services* (AWS) adicionam um grande valor às várias fases de desenvolvimento de um projeto de *software*. Permite aos seu clientes escolher e usar as ferramentas que mais se adequam ao desenvolvimento do seu projeto de *software*, de um vasto conjunto oferecido.

Permite também o desenvolvimento de vários tipos de aplicações, sejam elas *standalone*, *web* ou agentes automatizados.

O ponto mais forte da *Amazon Web Service* é sem dúvida o facto de que apenas são cobrados os recursos que são usados, além de que não são precisas infraestruturas com hardware e pessoal para administrar o mesmo, o que também diminui os gastos.

Sem dúvida que é um serviço muito promissor e com elevada potencialidade mas, ainda assim, a curva de aprendizagem poderá ser um obstáculo a uma maior utilização.

References

1. Bahga, A., Madisetti, V. (2013) *Cloud Computing: A Hands-On Approach*.
2. *Amazon Web Services* (2015). Disponível em http://aws.amazon.com/
3. *Amazon Simple Queue Service* (2015). Disponível em http://aws.amazon.com/sqs/
4. *Amazon DynamoDB* (2015). Disponível em http://aws.amazon.com/dynamodb/
5. *Amazon CloudSearch* (2015). Disponível em http://aws.amazon.com/cloudsearch/
6. *Amazon Glacier* (2015). Disponível em http://aws.amazon.com/glacier/