Initial Plan Improving responsiveness of autoscaling systems

By Gregory Nichols

Supervised by Omer Rana

CM3203 One Semester Individual Project, 40 credits

Project Description

Problem

At a company I am affiliated with there is a system implemented which manages the parsing jobs, mostly consisting of tweets to be analysed based on emotion. The system monitors the parsing queues and when the queue reaches above a certain threshold of messages the system automatically sends a request to the hosting company to create a new virtual machine based upon a specialised parsing image (which is an Ubuntu 14.04 image with the minimum resources to execute the parsing process). This image is manually created and uploaded to the hosting companies system every new patch. Once the queue has reduced in size the system recognises that there is less demand and downscales the computing power accordingly in order to reduce running costs.
While this system does indeed work it has several drawbacks;

- It is slow, there is a noticeable delay between the parsing queues reaching above the threshold and the parsers being created.
- It is inefficient as often the scaled up server will have too much computing power, leading to a sawtooth effect when it downscales.
- It is costly as creating a whole new server just for parsing has extra costs associated with it and the resources it requires to do so aren't used optimally.

Ideally any solution aims to alleviate these three issues as much as possible.

Context

The company I am doing this project for (Blurrt) is a social media analytics company, collecting social media posts from various sources (primarily twitter) and storing them appending them with scores based on their sentiment and emotion. The architecture consists of a collection system which offloads to a parsing queue, a parsing system which offloads to a storage queue and a storage system which stores data into an elastic cluster. There is a monitoring system which monitors the status of both queues. The front-end of the system pulls data as required through our API from the Elastic database to populate the web page.

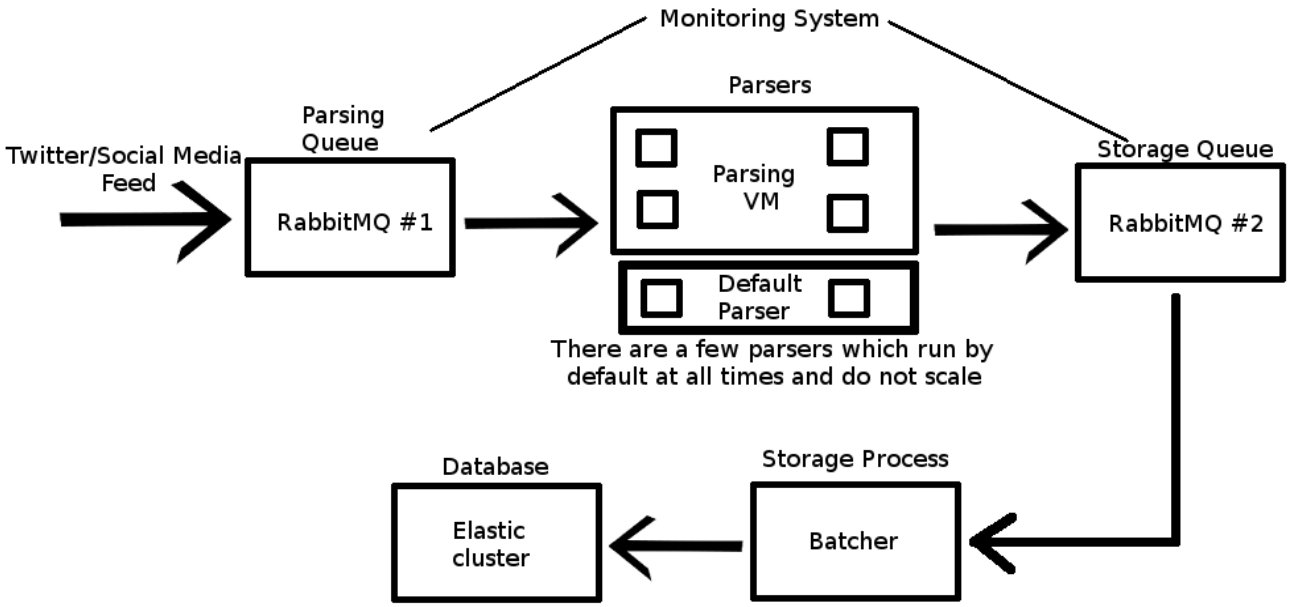

Diagram showing the architecture of the system from the incoming feed from collectors to the Elastic database

Overall Aims

The overall aims for this project will be to try and alleviate most if not all of the issues involved with the current system parsing system and put forward a new, quicker, less resource intensive and more scalable solution. My eventual solution should improve the deployment model in terms of reactivity, granularity and termination speed through better up and downscaling. It should also save the company money in running costs as well as be easier to patch. Another significant improvement I would like to implement would be a predictive algorithm/system in order to anticipate potential incoming workload on the system and scale up in advance in order to prevent the backlog ever getting too large. Finally as a potential extension if I wished to expand on these objectives further I could use the architectural improvements from upgrading the auto-scaling system across bottle necks in other areas such as API scaling on the front end.

Project Aims and Objectives

- Implement a system which is more reactive to backlogs of messages, meaning it is faster in terms of the speed it takes to scale up and scale down the parsing processes.
- Perform an analysis on the speed increase in time to create and destroy new parsers.
- Implement a system which is able to scale more granularly to prevent sawtoothing of the parsing queues.
- Implement a system which is easier to maintain, patching and installing updates are faster than the current system
- Implement a system which is cheaper to run over time than the current system, this may simply be due to there being less resources required or potentially from preventing there being wasted parsing resources when the queue is emptied.
- Implement a predictive algorithm to anticipate when there will be an incoming influx of tweets and account for it, scaling up accordingly to prevent backlogs occurring
- Apply this algorithm to try and predict when the demand is likely to drop and scale down accordingly to prevent extra resource usage.

Work Plan

	1	2	3	4	5	6	7	8	9	10	11	12
Write up detailed analysis of current system architecture	X	X										
Researching potential solutions for architecture (containers, scaling virtual machines)	X	X	X									
Research and write up a description of the current monitoring system and aspects which will need improving	X	X	X									
Research whether existing hosting solution will work with containers or scaling virtual machines & whether I will need alternative hosting	X	X	X									
Creating a detailed plan of the new system architecture		X	X									
Performing a performance analysis of the parser creation and destruction speed of the initial system		X	X	X								
Researching potential predictive algorithms			X	X	X							
Implementing the software solution			X	X	X	X	X	X	X			
Implementing the predictive algorithm					X	X	X	X	X			

Perform a performance analysis of the parser creation and destruction speed of the final system									X	X		
Write-up of results/Report									X	X	X	
Self Evaluation/Reflection											X	
Supervisor meetings			X			X			X			

Personal Milestones & Deliverables for final report

By the end of the second week I aim to have written an in-depth description and analysis of the current system's workings, making clear I properly understand of all aspects of the auto-scaling system.
By end of third week I aim to have researched potential technologies for the solution and write an analysis of how the current monitoring system works. I will also aim to write up what my new intended architecture will be and which technologies and hosting I will be using.
By the end of forth week I aim to have completed a performance analysis of the auto-scaling system writing up the current parser creation and destruction speed and documenting results, from which I can form comparisons later.
By the end of fifth week I aim to have researched potential predictive algorithms discussing in my meeting that week and in more detail during my longer meeting the week after which would be the best potential algorithm to choose.
By the end of week 9 I aim to have a demonstrable system working with the new architecture and a predictive algorithm working to monitor current status of the parsing queue.
By the end of week 10 I aim to have done a performance analysis of the new auto-scaling system, writing up the results of the parser creation and destruction speed and performing an in-depth comparison between that and the performance analysis of the original system.
By the end of week 11 I aim to have written up my report which will have been contributed to through the deliverables I created as part of my milestones as well as the documentation of work I do to further my goal. I also aim to complete my self evaluation and reflection discussing what I felt went well and what I have learnt from it and will take forward in future projects.