Auto-scaling Cloud Environments with Kubernetes

Felix Kollin

May 2019

1 Introduction

There are many cases when applications has to be able to handle unpredictable fluctuations in their traffic and thus, server load. A big problem many companies tackle is trying to predict these fluctuations and bracing the servers for the potential of heavy load.

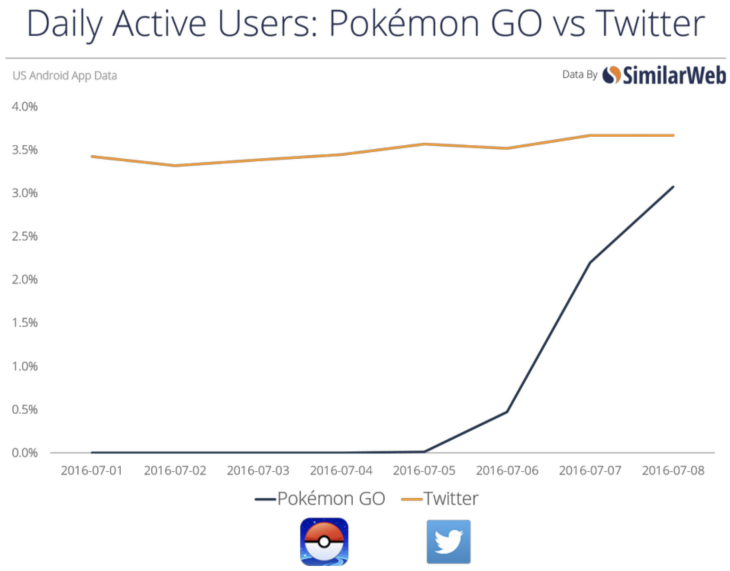

Figure 1: Daily active Android users of Pokémon Go compared to Twitter (Forbes.com, 2019).

However, they risk underestimating the traffic which may cause their service to become unavailable for an extended amount of time. This was the case with the popular Mobile game Pokémon GO during its launch in July of 2016. Their

original estimations were overthrown within the first hours of launch and the traffic eventually reached 50 times their estimation (Clouds Sky GmbH, 2019). The surge of popularity can be seen in Figure 1, where Pokémon go almost had as many daily active user as Twitter on Android. This resulted in the team having to rescale and redesign their architecture in a live scenario, something you can avoid by taking the necessary precautions. However, this is an extreme example, whereas most only need to account for some traffic variability.

The risk of being too careful and overestimating the traffic is that you may waste computational resources and thus suffer a loss due to traffic not meeting your expectations. Yet, Pokemon GO serves as a cautionary tale of how you should always "Build to Scale".

In order to adapt to workload fluctuations, most companies and services employ auto-scaling. With the use of auto-scaling, the workload determines the required resources and scales the application appropriately, both up and down. This has shown to a more cost-effective approach compared to proactive scaling while still maintaining Quality of Service (QoS) (Facebook Code, 2019. Rahman et al., 2018). There are several techniques and platform specific tools that can be used to scale cloud services. Some platforms which support auto-scaling include: Microsofts's Windows Azure, GCP (Google Cloud Platform), AWS (Amazon Web Services) and Oracle Cloud (Docs.microsoft.com, 2019. Amazon Web Services, Inc., 2019. Google Cloud, 2019. Oracle Help Center, 2019).

1.1 Scope and Purpose

This essay will primarily focus on Kubernetes specific technology because it is container management solution which can be installed on a variety of public and private clouds, such as the ones previously mentioned.

The goal of this essay is to investigate the auto-scaling techniques available in Kubernetes and the options available when employing auto-scaling with Kubernetes. Additionally, the essay will highlight some of the alternatives available and considerations when applying auto-scaling to projects. This aims to give the reader an overview of what to consider when employing auto-scaling by comparing the possible options.

1.2 Outline

The essay will provide a background section in order to introduce general terminology and techniques for auto-scaling. Following the background, the essay will cover Kubernetes specific approaches to auto-scaling, and how these techniques works within Kubernetes. A section with alternative approaches follows this section, which highlights the some alternative auto-scaling techniques. Then comes a section for discussing auto-scaling and the different techniques available. The essay concludes with the conslusion section, which highlights the main

points in this essay and concludes the discussion.

2 Background

There are several aspects of auto-scaling that are not specific to Kubernetes. Therefore, it's necessary to cover the general terms before covering Kubernetes specifics.

2.1 Horizontal Scaling

Horizontal scaling is often referred to scaling "out" or "wide". In cloud environments this means that the amount of replicated environments increase or decrease, with each one containing instances of your application. This requires orchestration as you will have to consider traffic entering any of the instances while still working as one single logical unit (Beaumont, 2019).

2.2 Vertical Scaling

Vertical scaling deals with workload fluctuations by allocating the required resources to an existing environment. In cloud environments this usually means the RAM or CPU dynamically scaling.

2.3 Scaling Techniques

2.3.1 Proactive Scaling

Scaling proactively is the traditional mindset which does not make dynamic scaling decisions, so it is not auto-scaling the environments at run time. Instead, the environment is scaled based on tests and previous experiences in a proactive way, which may result in resources not being utilized as they are not automatically scaled down when remaining unused.

2.3.2 Reactive and Conservative Auto-scaling

Both the reactive and conservative auto-scaling techniques consider current or past metrics in order to make scaling decisions. Reactive simply looks at the current time step, whereas conservative only makes decisions if the system deviates in the same way (below or above certain thresholds) in the last time steps within a set window. (Netto et al., 2014) Consequently, conservative is a type of reactive technique which waits and studies the system behavior before it makes a decision.

2.3.3 Predictive Auto-scaling

Predictive auto-scaling describes scaling techniques which aims to predict future workload fluctuations. This may be achieved by monitoring trends and applying

machine learning to make predictions. However, some simpler approaches makes predictions by only looking at the current time step (Netto et al., 2014).

3 Kubernetes Auto-scaling

The essence of auto-scaling with Kubernetes consists coordination of two layers of scalability. The first being the Pods layer autoscalers, which includes Horizontal Pod Autoscaler (HPA) and the Vertical Pod Autoscaler (VPA). They scale the available resources for containers. The second layer is on the cluster level, Cluster Autoscaler (CA), which scales the number of nodes in the cluster. Kubernetes provides a component that adjust the size of the Kubernetes Cluster so that all pods have a place to run and there are no unneeded nodes. However, the component is platform specific, but works with GCP, AWS and Azure.

The HPA in Kubernetes periodically checks metrics in order to determine if horizontal scaling should take place, thus applying a reactive scaling technique (Kubernetes.io, 2019). The user can configure which of the pre defined metrics the HPA should consider, such as the CPU or RAM. To avoid noise from starting and stopping pods affecting the metrics, scale-up can only occur once 3 minutes has passed since the last rescaling. Additionally, scale-down requires 5 a minute cooldown from the last rescaling. The period differs because rapid increase to account for a higher workload is prioritized to ensure QoS (GitHub, 2019).

$$desiredReplicas = \lceil currentReplicas(currentMetricValue/desiredMetricValue) \rceil \tag{1}$$

When deciding wether or not to scale, the essentially HPA operates on the ratio between the desired metric value and the current metric value, as shown in equation 1.

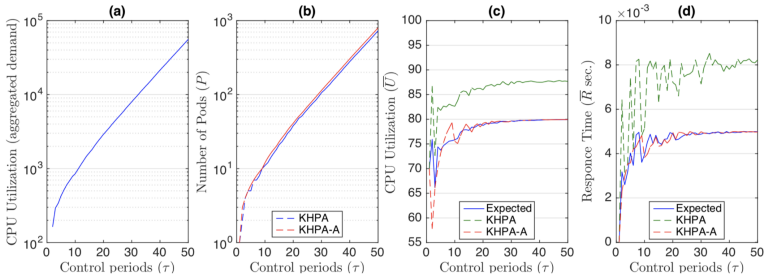

Figure 2: Comparison of the metrics for the relative KPHA (Kubernetes Horizontal Pod Auto-scaling algorithm) and the absolute version (KPHA-A) on how well they conform to the expected scaling policy. (Casalicchio et al., 2018).

Relative metrics are used to compute the required containers to keep the re-

source utilization below the specified threshold. Relative metrics measures the share that that each container has of the resources used, while its counterpart, absolute metrics, account for the utilization in the host system (VM or physical server). A study suggests that absolute metrics may reduce the application's response time, with partial results shown in Figure 2 (Casalicchio et al., 2018). The results indicate that the absolute metric results in better scaling in accordance to the expected CPU utilization policy while the relative metric may utilize more CPU. However, the user would need to guarantee that the target when scaling is lower than the requested resources. This can be troublesome as you would have to change the autoscaler utilization threshold each time the requested resources for a pod is changed (GitHub, 2019).

Kubernetes also supports vertical auto-scaling with their VPA. It is currently in Beta and can only be used together with a HPA if only operating on custom or external metrics.

4 Alternative Approaches

The main alternative to Kubernetes' reactive auto-scaling is predictive auto-scaling as it deals with some of the drawbacks of the reactive technique, something which this essay covers more in the next section, Discussion. Predictive auto-scaling has been covered in many studies due to the different prediction techniques which can be used. For example, one study investigates deep learning techniques, such as Recurrent Neural Network-Long Short Term Memory (RNN- LSTM) to predict the future workload (Radhika, 2018).

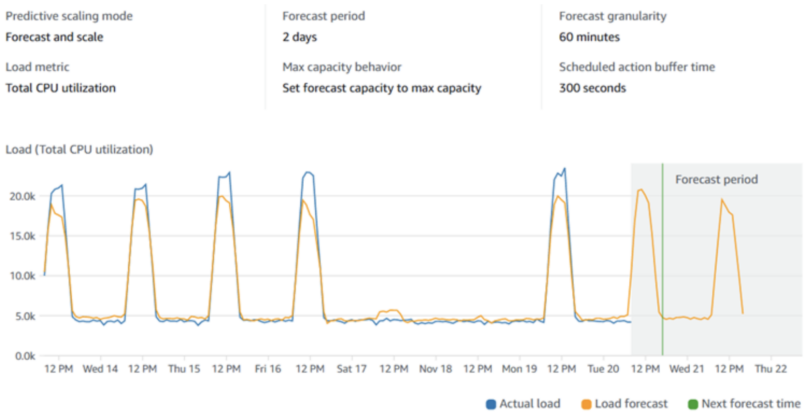

Figure 3: Amazon's predictive auto-scaling viewed from the AWS console (Amazon Web Services, 2019).

However, this may be hard to apply machine learning in practice for DevOps

engineers which are used to working with production ready tools, which is why Amazon has created their own predictive auto-scaling available for Amazon's ECS2 (Elastic Container Service 2) in AWS (Amazon Web Services, Inc., 2019). This can be configured directly and monitored in the AWS console, as seen in Figure 3 which shows how the predictive technique displays a forecast.

While it's difficult to investigate exactly how Amazon's predictive scaling works. It may be a hybrid approach, similar to Netflix's Scryer. Scryer employs a predictive technique but works in tandem with a reactive approach (Medium, 2019). In this case, the reactive part of the auto-scaling is used as a safety net in the cases where the prediction model failed to anticipate the workload.

5 Discussion

The first thing you need to consider looking into auto-scaling is your services' scaling capabilities. You may have to take scaling into account when developing the application. However, most of the time it's enough to ensure containerized execution compatibility and properly orchestrating micro services, which is a whole topic in itself. The exception to this is, for example, is if you're looking into using custom metrics. One example would be if you can easily predict when increased workload may occur from within your application, then a HPA would benefit from the added metric. Such metric could be queued tasks or requests/second, which could indicate that the workload is about to increase before the default metrics are able to detect it.

5.1 Horizontal or Vertical Auto-scaling

Your scaling needs may also differ depending on the scenario. If the CPU and RAM requirements differs between deployments and you are constantly adjusting the podspec, it may be worth considering VPA. Instead of employing a systematic process every time a service crashes due to lack of memory, the pod can scale vertically. Scaling only vertically can also be considered more simple because it does not require no changes in how your application functions, but it's limited as you cannot scale infinitely. Essentially, VPA scales based on your applications workload fluctuations, while HPA scales based on the overall service. A combination of HPA and VPA may be the most appealing, but it requires you to only feed custom metrics to the HPA (Google Cloud Blog, 2019).

5.2 Reactive or Predictive Auto-scaling

The nature of the workload fluctuations is the main factor when deciding between scaling strategies. More randomness makes predictive scaling struggle due to decreased accuracy, so it may miss some unplanned increases in workload. Whereas a workload with more bursts makes the predictive technique

overestimate the required resources. This can either be because of how the application works or the traffic, it is therefore worth considering that large bursts in computational requirements may also make the predictive approach suffer. Consequently, it's important to study all scaling metrics instead of just traffic or computational workload, which also applies to reactive thresholds.

If QoS is of most importance and you have occasional spikes in your workload, predictive may be the best choice as it does not suffer from the same start up delay as reactive. The start up delays may result in reactive decreasing QoS as the load is increased. This could be even more evident in Kubernetes with the 3 minute cooldown between up-scaling and the previous scaling action. A hybrid approach may be the best option in order to properly downscale after a workload burst. However, this is not a feature readily available and increases the overall complexity of the scaling technique, but has proven successful for Netflix.

5.3 Dealing With Complex Techniques

A complex auto-scaling technique may make it harder to adjust the metric thresholds. On the other hand, one patent filed by Amazon indicates that more accessible manual auto-scaling threshold adjustment is being worked on (Amazon Technologies, Inc., 2019). Hopefully this leads to a more streamlined adjustment process in the future.

5.4 Effects of Auto-scaling

Overall, with the right technique for your services, you can achieve the cost effectiveness observed by others once they started applying auto-scaling. Furthermore, it has environmental implications due to the decreased energy consumption. Therefore it aligns well with the mindset of leading companies, as the climate positive policy seems to grow more popular in the tech industry.

6 Conclusion

Kubernetes provides ample tools to employ auto-scaling across you project, but it requires you to analyze your services and adjust your scaling policies accordingly. Combining HPA with a VPA using custom metrics could be considered as the best approach, as it allows you to handle increased traffic by scaling the amount of pods, while it makes sure your application has enough resources. I believe VPA scaling is something much in line with what DevOps is all about, as it allow you to deploy in a more efficient manner without having to manually adjust the resource requests if the application's requirements have changed.

While Kubernetes provides reactive auto-scaling, predictive auto-scaling has

been observed to increase the overall QoS by applying machine learning techniques when scaling. However, while predictive auto-scaling can predict load spikes, it struggles with accuracy when your server load is of a random nature. Depending on the implementation, it may also result in less restrictive scaling on bursty workloads resulting in overestimation of required resources.

To conclude, the following message is left for the readers to stress the key points of this essay:

Do not employ auto-scaling without considering the implications and limitations of horizontal and vertical scaling detailed in this essay. Think about how your services operate and study the metrics that indicate scaling is require in order to determine if you should use a reactive or predictive auto-scaling technique.

References

Amazon Technologies , Inc. (2019). *VERSATILE AUTOSCALING FOR CONTAINERS*. US 2019 / 0089714 A1.

Amazon Web Services, Inc. (2019). *Introducing Predictive Scaling for Amazon EC2 in AWS Auto Scaling*. [online] Available at: https://aws.amazon.com/about-aws/whats-new/2018/11/ introducing-predictive-scaling-for-amazon-EC2-in-aws-auto-scaling/ [Accessed 12 May 2019].

Amazon Web Services. (2019). *Predictive Scaling for EC2, Powered by Machine Learning*. [online] Available at: https://aws.amazon.com/blogs/ aws/new-predictive-scaling-for-ec2-powered-by-machine-learning/ [Accessed 12 May 2019].

Beaumont, D. (2019). *How to explain vertical and horizontal scaling in the cloud - Cloud computing news*. [online] Cloud computing news. Available at: https://www.ibm.com/ blogs/cloud-computing/2014/04/09/explain-vertical-horizontal-scaling-cloud/ [Accessed 12 May 2019].

E. Casalicchio and V. Perciballi, "Auto-Scaling of Containers: The Impact of Relative and Absolute Metrics," 2017 *IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, Tucson, AZ, 2017, pp. 207-214.

Clouds Sky GmbH. (2019). *Pokémon GO as an example of what you can do with Kubernetes and Google Cloud Platform*. [online] Available at: https://cloudssky.com/en/blog/ Pokemon-GO-as-an-example-of-what-you-can-do-with-Kubernetes-and-Google-Cloud-Platform/ [Accessed 12 May 2019].

Docs.microsoft.com. (2019). *Autoscaling Guidance*. [online] Available at: https://

docs.microsoft.com/en-us/previous-versions/msp-n-p/dn589774(v=pandp.10) [Accessed 12 May 2019].

Facebook Code. (2019). *Making Facebook's software infrastructure more energy efficient with Autoscale - Facebook Code*. [online] Available at: https://code.fb.com/production-engineering/making-facebook-s-software-infrastructure-more-energy-efficient-with-autoscale/ [Accessed 12 May 2019].

Forbes.com. (2019). *'Pokémon Go' Is About To Surpass Twitter In Daily Active Users On Android*. [online] Available at: https://www.forbes.com/sites/jasonevangelho/2016/07/10/pokemon-go-about-to-surpass-twitter-in-daily-active-users/#2f8ccbee5d3e [Accessed 12 May 2019]. 8

GitHub. (2019). *kubernetes/community*. [online] Available at: https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/horizontal-pod- autoscaler.md [Accessed 12 May 2019].

GitHub. (2019). *kubernetes/community*. [online] Available at: https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/horizontal-pod- autoscaler.md [Accessed 12 May 2019].

Google Cloud Blog. (2019). *Using advanced Kubernetes autoscaling with Vertical Pod Autoscaler and Node Auto Provisioning*. [online] Available at: https://cloud.google.com/blog/products/containers-kubernetes/using-advanced-kubernetes-autoscaling-with-vertical-pod-autoscaler-and-node-auto-provisioning [Accessed 12 May 2019].

Google Cloud. (2019). *Autoscaling Groups of Instances*. [online] Available at: https://cloud.google.com/compute/docs/autoscaler/ [Accessed 12 May 2019].

Kubernetes.io. (2019). *Horizontal Pod Autoscaler*. [online] Available at: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ [Accessed 12 May 2019].

Medium. (2019). *Scryer: Netflix's Predictive Auto Scaling Engine*. [online] Available at: https:// medium.com/netflix-techblog/scryer-netflixs-predictive-auto-scaling-engine-a3f8fc922270 [Accessed 12 May 2019].

Netto, Marco & Cardonha, Carlos & Cunha, Renato & Assuncao, Marcos. (2014). *Evaluating Auto-scaling Strategies for Cloud Computing Environments*. 10.1109/MASCOTS.2014.32.

Oracle Help Center. (2019). *Administering PaaS Services*. [online] Available at: https:// docs.oracle.com/en/cloud/paas/psmon/automatic-scaling.html [Accessed 12 May 2019].

E. G. Radhika, G. Sudha Sadasivam and J. Fenila Naomi, "An Efficient Pre-

dictive technique to Autoscale the Resources for Web applications in Private cloud," *2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, Chennai, 2018, pp. 1-7.

Sabidur Rahman, Tanjila Ahmed, Sifat Ferdousi, Partha Bhaumik, Pulak Chowdhury, Massimo Tornatore, Goutam Das, Biswanath Mukherjee, "Dynamic Controller Deployment for Mixed-Grid Optical Networks", *Asia Communications and Photonics Conference (ACP) 2018*, pp. 1-2, 2018