Predictive Pod Autoscaling in the Kubernetes Container Cluster Manager

by

Matt McNaughton

Professor Jeannie Albrecht, Advisor

A thesis submitted in partial fulfillment
of the requirements for the
Degree of Bachelor of Arts with Honors
in Computer Science

Williams College
Williamstown, Massachusetts

October 31, 2015

DRAFT

Contents

Abstract

Acknowledgments

Chapter 1

Introduction

Over the past few decades, an explosion in the need for computing power and memory, and the existence of cheap, interconnected computers, has driven a significant increase in the feasibility and benefits of distributed systems. [8, pg. 1] First, we consider the origin of distributed systems as a field of computer science. Before the availability of cheap, powerful microprocessors and reliable, efficient local-area networks (LANs), computational tasks could only be performed on a singular computer. [8, pg. 1] If a task was too computationally expensive, the only solution was to run it on a larger, more powerful supercomputer. However, as cheap microprocessors made computers more abundant, and LANs allowed computers to communicate quickly, a new model of performing resource intensive computation, entitled **distributed systems** arose. In the distributed systems model a collection of individual computers function as a single computer to solve a given computing task. [8, pg. 2]

Second, we consider the ever-growing interest in unlocking and implementing the benefits of distributed systems. A number of forces drove, and continue to drive, increased interest in distributed systems over the past decade. The first, and most obvious, factor is the Internet, itself a distributed system. As more people connected to the Internet, through computers, mobile phones, and tablets, an increasing number of human interactions became computerized. Consumption, communication, research, and many more are all possible through the Internet. Naturally, large amounts of computing resources are needed to store the data, and perform the computation, related to these interactions. Closely coupled with this trend is the rise of "Big Data". In 2013, the digital universe contained 4.4 zettabytes of data.[1] [5] Naturally, without multiple computers working together it would be impossible to store and process this incredible volume of danger. Overall, it is nearly impossible to do anything in modern society without interacting with a distributed system and creating new digital data. Driving a car, trading a stock, visiting a doctor, checking an email, and even playing a simple video game, are all activities that distributed systems facilitate and improve. [6, pg. 4] As life becomes more and more computerized, and as the volume of data humans generate and hope to process grows and grows, distributed systems will only become increasingly more important. Furthermore, research into distributed systems makes it possible to continue to unlock, and make

[1] A zettabyte equals 10^{21} bytes, which equals 1 billion terabytes.

available to the general public, the incredible power of networked computers. As distributed systems supplying massive computational powers become more accessible, both because of decreased cost and increased ease of use, we can address an ever increasing number of challenging, important problems computationally.

There are a number of different models of computing tasks requiring high levels of computing resources:

1. Supercomputing: The supercomputing model responds to increased demands for computing resources by increasing the specs of the computer far beyond the range of the traditional commodity computer. While supercomputers are able to avoid the majority of the complications the introduction of networks introduce, most prominently reliability and security, there are naturally limits on the power of supercomputers. Importantly, constructing supercomputers is extremely expensive, and thus their computing power is not available to the general public. Furthermore, it is difficult to scale a supercomputer should the need arise. Finally, supercomputers offer a single point of failure, meaning they are not particularly robust to error. These limitations have decreased the usage of supercomputers to provide the mass of computing power needed in the "Big Data" era.

2. Cluster Computing: Cluster computing is defined as utilizing "a collection of similar workstations of PCs, losely connected by means of a high-speed local-area network [where] each node runs the same operating system." [8, pg. 17-18] Cluster computing can provide a mass of computing power similar to that contained in a supercomputer. Cluster computing also offers many advantages over the single supercomputer. First, and perhaps most importantly, they are much more cost-efficient. Second, clusters are easy to scale by simply adding new nodes. Finally, cluster computing is much more fault tolerant, as a singular failing commodity computer will simply be removed from the cluster. Cluster computing is used in the implementation of what is colloquially referred to as *Cloud computing*, in which large amounts of computing resources are offered on a per-usage basis. [6, pg. 13] Cloud computing, as implemented by Amazon Web Services, [1] Microsoft Azure, [4] and Google Compute Engine, [2] continue to revolutionize the development and deployment of computing applications, as developers gain access to cheap, easily accessible, quickly scalable computing power.

3. Grid Computing: Grid computing is similar to concept in cluster computing, except it foregoes the requirement that all computers within the grid be homogeneous. As such, the grid computing model accounts for a large degree of heterogeneity with respect to network membership, operating system, hardware, and more. [8, pg. 18] While grid computing systems weaken the homogeneity requirements imposed by cluster computing, the resulting heterogeneity introduces significant complexity.

In this thesis, we focus on cluster computing. Specifically, we focus on the cluster manager, an integral component of cluster computing. Cluster managers are responsible for abstracting all of the management details of the distinct nodes in the cluster, and instead present a single mass of computing resources to the user on which they can run jobs on the application. In other words, a

cluster manager "admits, schedules, starts, restarts, and monitors the full range of applications" run on the cluster. [10, pg. 1] There are a variety of different cluster managers, the most important of which will be discussed in the background chapter, each pursuing different objectives. This thesis will ultimately focus on **Kubernetes**, an open-source cluster manager from Google. [3].

Cluster managers seek to accomplish a number of different goals, and as a result, multiple metrics indicate success. For example, Microsoft's Autopilot is predominantly concerned with application uptime, and thus success is measured with respect to reliability and downtime. [7, pg. 1] Alternatively, a number of cluster managers measure themselves based on efficient resource usage. [10, pg. 7] Essentially, efficient cluster resource utilization relates to the percent of cluster resources which are actually being used. For example, one such measurement of this metric, cluster compaction, examines how many computers could be removed from the cluster, while still comfortably running the cluster's current application load. [9, pg. 5] This metric is particularly important, because the more efficient the cluster management is at utilizing resources, the less resources cost, and the more accessible cluster computing becomes to the general public. A final important cluster management metric is quality of service. Quality of service measures the ability of an application to function at a specified level, despite ever-changing external factors. Again, this metric is particularly important, because increasing the robustness of applications running on cluster managers means these applications can be trusted with increasingly important tasks. Cluster managers predominantly differ with respect to which of these metrics optimize for, and the process by which this optimization occurs.

1.1 Goals

This thesis is most concerned with maximizing the efficient resource utilization and quality of service metrics with respect to the Kubernetes cluster manager. As such, this thesis pursues three goals:

1. Given an application running on a Kubernetes cluster, we seek to determine a method which ensures quality of service stays consistently high regardless of external factors. While it is difficult to make guarantees regarding quality of service, it is possible to ensure each application has, and is utilizing, the resources it needs to function property. Given the cluster manager grants the application the resources it needs to function given the current external load, the cluster manager has done all it can to ensure application quality of service.

2. Given a certain number of applications running on a Kubernetes cluster, we seek to determine a method which ensures the cluster is as small as possible, while still comfortably supporting the application's current, and future, resource needs.

3. Given Kubernetes is an open-source project, we seek to implement, test, and evaluate a proposed enactment of the previous two goals. Thus, the methods we pursue will in part be dictated by the current structure and implementation of Kubernetes. Tests be conducted using the Google Compute Engine [2] and both simulated and real Kubernetes user data.

1.2 Contributions

This thesis presents our given contributions to Kubernetes. Kubernetes seeks to ensure high application quality of service and efficient use of computing resources, and our contributions look to further its ability to accomplish these goals. As such, we present not only new methodology, but also new, working implementations, with accompanying evaluation. We demonstrate the effectiveness of our modifications in comparison to the non-modified Kubernetes using both simulated and real-world datasets. Finally, we discuss the experiences of making these modifications on Kubernetes, as well as the avenues for future improvements with respect to Kubernetes and cluster managers in general.

1.3 Contents

@TODO.

Chapter 2

Background

Bibliography

[1] Amazon web services. https://aws.amazon.com/.

[2] Google compute engine. https://cloud.google.com/compute/.

[3] Kubernetes website. http://kubernetes.io.

[4] Microsoft azure. https://azure.microsoft.com/en-us/.

[5] Corporation, I. D. The digital universe of opportunities: Rich data and the increasing value of the internet of things, 2014.

[6] Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. *Distributed Systems: Concepts and Design*, 5th ed. Addison-Wesley Publishing Company, USA, 2011.

[7] Isard, M. Autopilot: Automatic data center management. *SIGOPS Oper. Syst. Rev. 41*, 2 (Apr. 2007), 60–67.

[8] Tanenbaum, A. S., and Steen, M. v. *Distributed Systems: Principles and Paradigms (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[9] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems* (New York, NY, USA, 2015), EuroSys '15, ACM, pp. 18:1–18:17.

[10] Verma, A., Pedrosa, L., Korupolu, M. R., Oppenheimer, D., Tune, E., and Wilkes, J. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems (EuroSys)* (Bordeaux, France, 2015).