Implentation of an simplistic Interface for Big Data Workload Scheduler in Kubernetes

Implentation of an simplistic Interface for Big Data Workload Scheduler in Kubernetes

Lukas Schwerdtfeger

A thesis submitted to the
Faculty of Electrical Engineering and Computer Science
of the
Technical University of Berlin
in partial fulfillment of the requirements for the degree
Bachelor Technische Informatik

Berlin, Germany
December 22, 2015

Technische
Universität
Berlin
berlin

Main supervisor:

Prof. Dr. habil. Odej Kao, Technical University of Berlin

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den

Zusammenfassung

Kurze Zusammenfassung der Arbeit in 250 Wörtern.

Abstract

Short version of the thesis in 250 words.

Acknowledgements

This chapter is optional. First of all, I would like to...

Contents

1 Introduction **1**
1.1 What is Big Data? 1
1.2 Why is Scheduling Important 1

2 Background **3**
2.1 What is Spark Batch Frameworks? 3
2.2 What is Kubernetes? 3

3 Related Work **5**
3.1 Scheduler 5
3.2 Scheduler Frameworks 5

4 Approach **7**
4.1 Deep Dive: Batch-Job Operator 7
4.2 Deep Dive: Kubernetes Scheduler 7
4.3 Deep Dive: Extender 7
4.3.1 Four ways to extend Kubernetes Scheduler 7
4.3.2 How the scheduler extender works 8
4.3.3 What is an Extender 8

5 Evaluation **9**
5.1 Does it Work? 9
5.2 Comparision between different Apporoaches 9

6 Open Questions **11**
6.1 Future Work for the Operator 11

7 Conclusion **13**

List of Figures

List of Tables

1 Introduction

Short section to give a brief overview into the topics of big data and scheduling before diving deeper.

1.1 What is Big Data?

This section should provide some Motivation for the Topic of BigData

- ◇ Companies collecting more and more data
- ◇ Learn from Data to provide better products / customer experience
- ◇ DataLakes and the sheer amount of volume of data

1.2 Why is Scheduling Important

This section should explain, what makes scheduling an important role in large scale data processing.

2
Background

This section will be a short introduction into the background chapter and explain why the chapter is further split into BatchJob frameworks and Kubernetes

2.1 What is Spark Batch Frameworks?

This Section will give an Introduction to Distributed Big Data Processing and the Value of already Existing Big Data Frameworks like Spark and Flink

2.2 What is Kubernetes?

This Section should give the Reader a Background to Kubernetes and maybe other Cluster Management Systems like YARN and compare them. Also this should explain why this work is only Focused on Kubernetes

3
Related Work

Explain that both kubernetes and big data scheduling are huge topics with lots of related work

3.1 Scheduler

pick some related work in regards of big data scheduling

- Paragon, Quasar
- Introduction to Mary and Hugo

3.2 Scheduler Frameworks

pick some related work in regards of scheduling in kubernetes

- Kubernetes Scheduling Framework
- Volcano

4 Approach

During the initial phase of collecting Resources regarding influencing the Kubernetes Scheduler, I discovered multiple ways I could accomplish my goals. This section shall give a brief overview of the different Approaches i could have chosen, and maybe provide an answer on why i did not chose them in the end.

- ◇ Implementation of Scheduling Algorithm, like Hugo and Mary directly inside the Kube-Scheduler, either by replacing the original or extending with a plugin
- ◇ Implementation of the Interface directly inside the Kube Scheduler, forgoing the Operator/Controller
- ◇ Implementation of the Interface as an Extender

4.1 Deep Dive: Batch-Job Operator

Give a brief introduction how the Operator Pattern in kubernetes is used Show the Operator Framework i used Explain hurdles I encountered implementing the Operator Explain its functions

4.2 Deep Dive: Kubernetes Scheduler

Explain how the Kubernetes Scheduler works Describe how an Operating Cycle works Explain at which points we need to interact with the Scheduler

4.3 Deep Dive: Extender

https://developer.ibm.com/articles/creating-a-custom-kube-scheduler/

4.3.1 Four ways to extend Kubernetes Scheduler

- ◇ Modify Upstream Source code, recompile and run scheduler

- ◇ Separate Scheduler, that runs a long (scales bad, such as a distributed lock and cache synchronization)
- ◇ scheduler extender
- ◇ scheduler framework

4.3.2 How the scheduler extender works

How does the scheduler function? - scheduler starts with parameters give - watches pods with empty .spec.nodeName puts them in a queue - pops out a pod from the queue and starts scheduling cycle - filter (based on hard requirements, etc) - score (soft requirement) - bind (api server)

4.3.3 What is an Extender

Extender is a set of API endpoints - /filter Filter based on extender-implemented predicate functions. The filtered list is expected to be a subset of the supplied list. The failedNodes and failedAndUnresolvableNodes optionally contains the list of failed nodes and failure reasons, except nodes in the latter are unresolvable. - Arguments: Pod + List of Nodes - Expects: List of Nodes (subset of Input Nodes) - /prioritize Prioritize based on extender-implemented priority functions. The returned scores & weight are used to compute the weighted score for an extender. The weighted scores are added to the scores computed by Kubernetes scheduler. The total scores are used to do the host selection. - Arguments: Pod + List of Nodes - Expects: List of Nodes with Scores + Weight - /bind Delegates binding - Arguments: Binding (which is essentially Pod + TargetNode) - Expects: Error if occurred - /preempt ProcessPreemption returns nodes with their victim pods processed by extender based on given: 1. Pod to schedule 2. Candidate nodes and victim pods (nodeNameToVictims) generated by the previous scheduling process. The possible changes made by extender may include:

5 Evaluation

5.1 Does it Work?

Should give a brief summary of the functionality that the operator is supporting in the current state. Show some stats collected using the operator and running multiple BatchJobs on kubernetes

5.2 Comparision between different Apporoaches

This Section should show the comparison between the different approaches of extending the Kubernetes Scheduler and show the value of having a simplifying layer of abstraction above already existing kubernetes scheduler frameworks, by presenting the ease of use of an external already implemented scheduling algorithm like Mary.

6
Open Questions

Maybe this section is overkill

6.1 Future Work for the Operator

Give a brief overview of what I belief is missing in the current implementation of the Scheduler Interface

7 Conclusion

This Section should be the final conclusion

Bibliography