Experimental Study on Scalable Post-Groomer of the Wildfire HTAP System on Cloud

Project Report

from the Course of Studies Applied Computer Science

at the Cooperative State University Baden-Württemberg Mannheim

by

Pascal Schroeder

09/24/2018

Time of Project	05/07/2018 - 08/24/2018
Student ID, Course	5501463, TINF16AI-BC
Division, Business Unit	IBM Research
Company	IBM Deutschland GmbH, San Jose - Almaden
Supervisor in the Company	Yuanyuan Tian

Signature Supervisor _____

Contents

Declaration of Sincerity

Hereby I solemnly declare that my project report, titled *Experimental Study on Scalable Post-Groomer of the Wildfire HTAP System on Cloud* is independently authored and no sources other than those specified have been used.
I also declare that the submitted electronic version matches the printed version.

Mannheim, 09/24/2018
Place, Date

Signature Pascal Schroeder

Abstract

1 Introduction

1.1 Advantage of cluster systems

Cluster systems are two or more computers connected to each other, so that they can share their resources and can be viewed as one system. Therefore they can be connected physical as well as virtual. Using such a cluster system is typically much more cost-efficient compared to using a single computer, which provides about the same power. But the advantage of such systems doesn't end with a better cost-efficiency.

First clusters are not only used for one specific purpose, but there are different kinds of cluster configurations, which are build for different objectives.

On the one hand, there are "Load balancing" cluster configurations. Load balancing clusters are clusters with the objective to provide better computing performance, like minimizing response time and maximizing throughput. Therefore it splits the computation in different parts and runs them parallel on different nodes. Through that the capabilities of the systems can be combined.

Furthermore there are "High availability" clusters, which are designed to prevent a total breakout of the system. For that there are several redundant nodes connected to the system. That means, that when one component fails, a redundant node of this component is used to provide the service. Thereby availability is ensured even if one component fails. The more redundant nodes are used the better will be the availability of a system. To provide an example: If a system has an availability of 90% it has a downtime of 10%, which means that the system is not available on 36.53 days per year. If there is a second, redundant and independent node connected to the cluster, which acts as a stand-in for the first system in case of a failure, the downtime of those two systems has to be combined. That means that this cluster would only fail if there is an overlap between the downtimes of those two systems. Resulting of a downtime of 10% of each system there is a combined downtime of 1% or 3.65 days per year. If there is a third system connected to the cluster, the downtime will be reduced to 0.1% or 8.77 hours per year and so on. The job of "High availability" clusters is in that to reduce the downtime of the system and to provide a working system almost any time.

Figure 1.1 shows an example architecture of such a "High availability" cluster. There are two servers, whereby one server is a replication of the other one. The request router

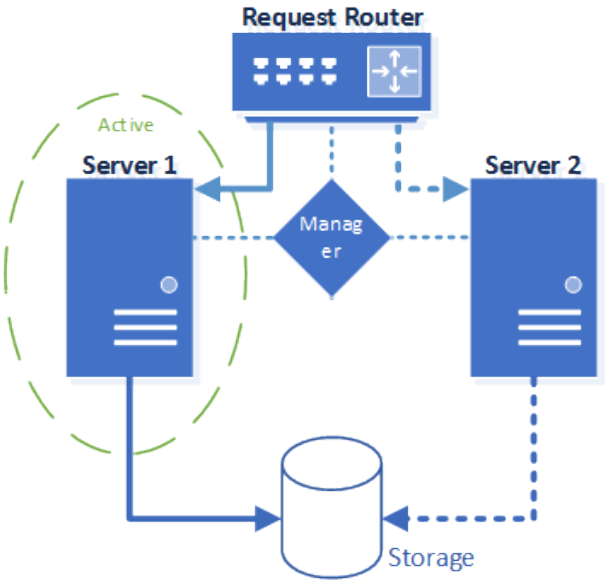

Figure 1.1.1 “High availability” cluster architecture

handles the incoming requests and leads them to the active server. If there is a system failure of the active server, the manager will recognize that, and sends a message to the request router to change the request address to the other server. From now on this server will deal with all the request. Also there is a shared storage for both of the servers in one database. Whether which one is active they can both access this database.

But those advantages don’t have to stand alone, but they can also be combined, so that you can provide high availability as well as high performance. Those two problems are becoming more and more important because of the high amount of calculations which are made in Machine Learning, Big Data Analysis etc. That’s the reason why cluster systems are becoming more and more important in modern IT for providing a solution for both problems.

1.2 Different kinds of data processing system

In the modern times of machine learning and Big Data analysis the importance of data processing systems increases continuously. More and more systems are being released for Big Data Analysis, for example Spark. Such systems are called OLAP - Online Analytical Processing. One example OLAP system is Spark. Such systems are working with groups of data and complex requests. For better analyses it also doesn’t only use the current state of a dataset, but also its history. The objective of OLAP systems is to give one unique, consistent view on all data from different sources. and extracting information like prediction or other complex information out of these data.

But still conventional data processing like ACID Atomicity, Consistency, Isolation, Durability transactions is needed for regular database requests, like bank transactions or text messages. Those systems are called Online Transactional Processing - Online Transactional Processing. This system is characterized by instant instant interactions with the business systems without a latecy instead of collecting all transactions and process them all at once in the night. This was a major step forward in fast data processing in the late 1970s and those systems are still used by almost all companies.

The problem ist, that both of these systems need a specific form of their data, so they can't work together on the same set of data. For resolving that problem HTAP (Hybrid Transactional Analytical Processing) has been created. This enables real-time analytical results, which can help businesses in quick decision making. Therefore are still two copies of the data needed, but with an highly increased synchronization rate between those two data copies.

The Wildfire system, which was developed at IBM Research Lab Almaden, is such an HTAP system. It used Spark as OLAP engine and an own engine for OLTP. In the following chapters this system and underlying basement technologies will be presented. Furthermore an experimental study will be described, in which a part of this solution will be deployed on a Cluster system and tested concerning its performance.

1.3 Introduction to IBM Research Cloud

For a few years there has been a trend to move from local, physical devices to centralized cloud platforms. This has different advantages like dynamically allocation of computing power and more cost-efficiency. The reason for this is because often there are unused resources on physical devices, because most systems don't need all of their power anytime. Through outsourcing this computing power on the cloud the resources are only used where they are needed. This increases the cost efficiently eminent, because not every physical device needs backup resources for situations, in which more computing power is needed. Instead with the cloud it is possible to request more resources when needed and to release them when they are not needed any longer.

In Research there is often a lot computing power necessary for testing, calculating and building projects and experiments. Therefore every labour has several own physical devices. But currently there is also a trend in Research projects to move to Cloud, sharing resources and save costs. That is the reason why IBM started its "IBM Research Cloud", hosted on iRIS (IBM Research Integrated Solutions) IMS (Intrastructure Management System).

This is a self-service platform, which provides cloud based solutions for IBM Researcher on request for specific resources.

IBM Research Cloud itself provides virtual devices as well as virtual GPU (Graphical Processor Unit devices. Virtual devices mimics physical hardware but without the need of an isolated physical machine. Instead its just splitting some computing power of a big cluster via Software and makes the system believe, that it would be an isolated hardware device. Virtual GPU devices are differentiated by an additional Graphical Unit, which provides the virtual device even more computing power and relives the CPU (Central Processing Unit) in some tasks.

Besides the creation and deletion of virtual devices, the Research Cloud also provides operations like hard/soft reset or rebooting a device. Additionally it also provides the opportunity to create an image of an existing virtual device and creating a new one from this image. Through that you can keep your configurations and files from your old device and create a device based on those configurations without the need of installing everything up from the beginning.

In the following described project this Research Cloud will be used to setup a Kubernetes Cluster with several virtual devices and to run Wildfire using this Cluster.

1.4 Project objective

2 Theory

2.1 Kubernetes cluster solution

One very common system for managing cluster systems as described in chapter 1.1 is Kubernetes - or short K8s. Originally developed by Google and by now maintained by the Cloud Native Computing Foundation, Kubernetes is an “open-source platform for managing containerized workloads and services”. For unterstanding what that means the concept of containers needs to be described first.

Containers are isolated, stand-alone packages of software, similar to processes. In those packages everything is included, which this piece of software needs, like runtime, libraries, settings and other system tools. These containers have a completely different environment within themselves than outside. This environment included for example network routes, dns settings and control group limits. This enables the possibility to share common resources and still be isolated from any other process as well as the host system. Thereby containers are always working the same, no matter on what system they run or in which environment.

Kubernetes is for an automating deployment, scaling and management of these containers within a cluster of nodes. Thereby there has to be at least one master node. This master nodes owns the API-Server, the scheduler, the controller manager and a key value store called etcd.

First a pod is the smallest unit in Kubernetes. It contains one ore more containers, which are deployed together on the same host. Thereby they can work together to perform a set of tasks.

The clients uses the API Server to run all of their requests against it. That means the API Server is responsible for the communication between Master and Worker nodes and for updating corresponding objected in the etcd. Also the authentication and authorization is task of the API Server. The protocol for the communication is written in **REST!** (**REST!**). For reacting on changes of clients there is also a watch mechanism implemented, which triggers an action after a specific action, like the scheduler creating a new pod.

The Controller Manager is a daemon, which embeds all of the Kubernetes controller. Examples for them are the Replication Controller or the Endpoint Controller. Those controllers are watching the state of the cluster through the API Server. Whenever a

specific action happens, it performs the necessary actions to hold the current state or to move the cluster towards the desired state.

The scheduler manages the binding of pods to nodes. Therefore it watches for new deployments as well as for old ones to create new pods if a new deployment is created or recreating a pod whenever a pod gets destroyed for some reason. The scheduler organizes the allocation of the pods within the cluster on the basis of available ressources of the pods. That means, that it always create pods, where the most resources are available, or reorganize the allocation if there is a change in the resource allocation of the cluster.

The etcd is a key-value store, which stores the configuration data and the condition of the Kubernetes cluster. The etcd also contains a watch feature, which listens to changes to keys and triggers the API server to perform all necessary actions to move the current state of the cluster towards the desired state.

2.2 Spark

2.3 Wildfire

3 Method

3.1 Catalogue of Criteria

3.2 Configuration of Kubernetes cluster

3.3 Running Wildfire post-groomer on cluster

3.4 Experimental tests with the post-groomer in Wildfire

4 Result

4.1 Possibilities to deploy Wildfire post-groomer on Kubernetes cluster

4.2 Results of performance tests

5 Discussion

5.1 Evaluation of advantages on cloud-based usage of Wildfire

5.2 Evaluation of Wildfire's post-groomer performance on cluster

Acronyms

ACID Atomicity, Consistency, Isolation, Durability

OLAP Online Analytical Processing

Spark Apache Spark - Open Source Software for Big Data Analysis

OLTP Online Transactional Processing

HTAP Hybrid Transactional Analytical Processing

iRIS IBM Research Integrated Solutions

IMS Intrastructure Management System

GPU Graphical Processor Unit

CPU Central Processing Unit

K8s Kubernetes

Bibliography