2016 International Conference on Platform Technology and Service (PlatCon)

Proceedings

15-17 February 2016
Jeju, Korea

IEEE

IEEE Catalog Number: CFP16F03-ART (Xplore)
ISBN: 978-1-4673-8685-2 (Xplore)

IEEE Catalog Number: CFP16F03-CDR (CD)
ISBN: 978-1-4673-8684-5 (CD)

Twister Platform for MapReduce Applications on a Docker Container

Yunhee Kang
Division of Information and Communication
Baekseok University
115 Anseo Dong, Cheonan, 31065, Korea
yunh.kang@gmail.com

R. Young Chul Kim
SELab., Dept. of Computer Information Communications
Hongik University
2639, Sejong Campus, 30016, Korea
bob@hongik.ac.kr

***Abstract*—Docker is one of ways to provide more light-weight for agile computing resource based on container technique to handle this problem. For this work we have chosen this specific tool due to the increasing popularity of MapReduce and cloud container technologies such as Docker. This paper aims at automatically configuring Twister workloads for container-driven clouds. Basically this is the first attempt towards automatic configuration of Twister jobs on container-based cloud platform for many workloads.**

***Keywords*—Docker; agile computing; MapReduce; Twister**

I. Introduction

When prototyping a distributed application like MapReduce, a developer needs both to ensure the application execution corresponds to the specification and that its performance is not impacted by the number of nodes or by some failure scenarios [1-3]. Indeed, MapReduce relies on successive computing-commination steps that, if not coordinated with care, lead to performance bottlenecks and a poor scalability. However, this technology is both grounded on extremely complex platforms that are often difficult to understand, configure and optimize details.

As in rare situations both objectives can be reached at once, it is usual to start prototyping in a small set of nodes and then, when the execution was proved, perform additional scalability and fault tolerance tests.

Because all these steps require the execution of well-defined scenarios, we decided to rely on container-based virtualization, which allow the researchers to control both system images and network interconnections. The configuration can have a profound impact on application performance, security as well as availability, and the values they provide to end-users. Clearly, making these configurations manually is cumbersome or even impractical.

For this work we have chosen this specific tool due to the increasing popularity of MapReduce and cloud container technologies such as Docker. Basically this is the first attempt towards automatic configuration of Twister jobs on container-based cloud platform VM for many workloads. This paper aims at automatically configuring Twister workloads for container-driven clouds. Basically this is the first attempt towards automatic configuration of Twister jobs on container-based cloud platform VM for many workloads.

II. Related Works

A. Twister

Twister is one of MapReduce implementations, which is an enhanced MapReduce runtime with an extended programming model that supports an iterative MapReduce computing efficiently [4-5]. In addition it provides programming extensions to MapReduce with broadcast and scatter type for transferring data. These improvements allow Twister to support iterative MapReduce computations highly efficiently compared to other MapReduce runtimes. The demanding requirements have led to the development of a new programming model like Twister based MapReduce. It reads data from local disks of the worker nodes and handles the intermediate data in distributed memory of the worker nodes.

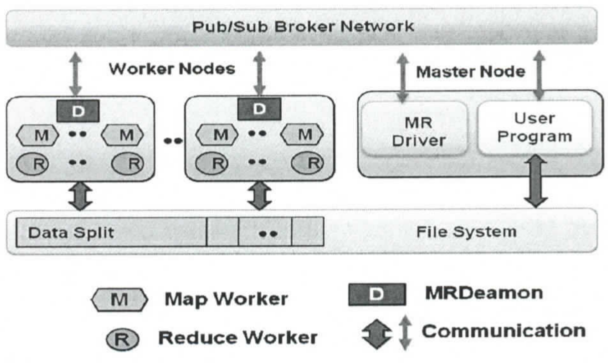


Fig. 1. Overall architecture of Twister

As shown in Figure 1, all communication and data transfers are performed via a pub/sub broker network via NaradaBrokering that is an open-source, distributed messaging infrastructure [9]. Twister uses a pub/sub messaging infrastructure to handle four types of communication needs; (i) sending/receiving control events, (ii) sending data from the client side driver to the Twister daemons, (iii) transferring intermediate data between map and reduce tasks, and (iv) sending the outputs of the reduce tasks back to the client side driver to invoke the combine operation.

B. Docker

Virtualization provides a way to abstract the hardware and system resources from an operation system, which is used to reduce the actual number of physical servers and to improve scalability and workloads in the cloud environment. In cloud computing environment, a VM is a computing platform that creates a virtualized layer between the computing hardware and the application.

Docker is an open platform released as a container project, originated by dotCloud, for developers and system administrators to build, ship and run distributed applications. It automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating-system-level virtualization on Linux and enables applications to be quickly assembled from components and eliminated the friction. Especially virtualization technology is the key technology, though it, deployment of cloud computing system can be implemented.

III. Experimental Results

A. K-means clustering

K-means clustering is a method of cluster analysis, which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. Given a data set, D, of n objects, and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions (k <= n), where each partition represents a cluster. The error function used is the sum of the distance that each point is from its cluster's centroid. In the k-means MapReduce application we built, it is used to determine the number of iterations in the data set. We describe an overall framework of a MapReduce based k-means with an initial starting configuration of this k-means clustering as input parameters. Figure 2 shows the designed framework.

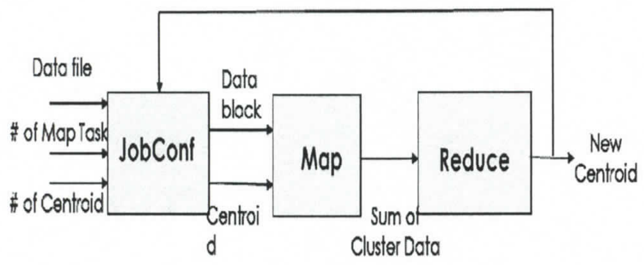
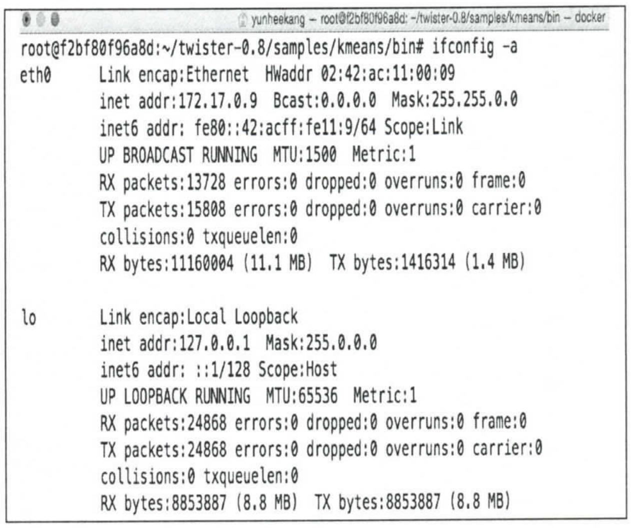


Fig. 2. Designed framework of a MapReduce based k-means

B. Experiment Environment

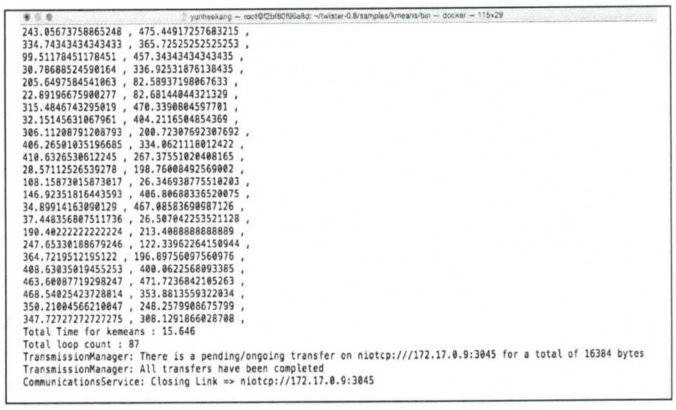
The following shows the two Docker containers generated from the image ubuntu:twister. Each of these containers has a network interface with IP addresses 172.17.0.9 and 172.17.0.10 respectively.



```
yunheekang — root@f2bf80f96a8d: ~/twister-0.8/samples/kmeans/bin — docker
root@f2bf80f96a8d:~/twister-0.8/samples/kmeans/bin# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 02:42:ac:11:00:09
          inet addr:172.17.0.9  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:9/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:13728 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15808 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:11160004 (11.1 MB)  TX bytes:1416314 (1.4 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:24868 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24868 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8853887 (8.8 MB)  TX bytes:8853887 (8.8 MB)
```

The following shows the result of k-means MapReduce application. To evaluate performance of a k-means MapReduce application, we define a configuration of this experiment. This configuration is used to cluster 3,000 data points with 3 mapper tasks and 1 reducer task. The elapsed time is 15.646 second with 87 iterations. In the perspective of elapsed time, there is no difference between physical machine and virtual machine.



```
yunheekang — root@f2bf80f96a8d: ~/twister-0.8/samples/kmeans/bin — docker — 115×29
243.05673758865248 , 475.44917257683215 ,
334.74343434343433 , 365.72525252525253 ,
99.51178451178451 , 457.34343434343435 ,
30.78688524590164 , 336.92531876138435 ,
205.6497584541063 , 82.58937198067633 ,
22.89196675900277 , 82.68144044321329 ,
315.4846743295019 , 470.3390804597701 ,
32.15145631067961 , 404.2116504854369 ,
306.11208791208793 , 200.72307692307692 ,
406.26501035196685 , 334.0621118012422 ,
410.6326530612245 , 267.37551020408165 ,
28.57112526539278 , 198.76008492569002 ,
108.15873015873017 , 26.346938775510203 ,
146.92351816443593 , 406.80688336520075 ,
34.89914163090129 , 467.08583690987126 ,
37.448356807511736 , 26.507042253521128 ,
190.40222222222224 , 213.40888888888889 ,
247.65330188679246 , 122.33962264150944 ,
364.7219512195122 , 196.89756097560976 ,
408.63035019455253 , 400.0622568093385 ,
463.60087719298247 , 471.7236842105263 ,
468.54025423728814 , 353.8813559322034 ,
350.21004566210047 , 248.2579908675799 ,
347.72727272727275 , 308.1291866028708 ,
Total Time for kemeans : 15.646
Total loop count : 87
TransmissionManager: There is a pending/ongoing transfer on niotcp:///172.17.0.9:3045 for a total of 16384 bytes
TransmissionManager: All transfers have been completed
CommunicationsService: Closing Link => niotcp://172.17.0.9:3045
```

IV. Conclusions

This paper described the overall process of building the experimental environment for Twister applications. For this work we have chosen this specific tool due to the increasing popularity of MapReduce and cloud container technologies such as Docker. Basically this is the first attempt towards automatic configuration of Twister jobs on container-based cloud platform VM for many workloads. This paper aims at automatically configuring Twister workloads for container-driven clouds. Basically this is the first attempt towards automatic configuration of Twister jobs on container-based cloud platform VM for many workloads. In the perspective of elapsed time, there is no difference between physical machine and virtual machine.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011601) and the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (NRF-2015H1C1A1035548).

REFERENCES

[1] Dean, J., Ghemawat, S.: MapReduce: A Flexible Data Processing Tool. CACM 53, 72-77 (2010)

[2] Morton, K., Friesen, A., Balazinska, M., Grossman, D.: Estimating the Progress of MapReduce Pipelines. IEEE 26th International Conference on Data Engineering (ICDE), 2010, pp. 681 - 684, Long Beach, CA (2010)

[3] Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. CACM 51, 107-113 (2008)

[4] Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., Fox, G.: Twister: A Runtime for Iterative MapReduce. The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) - HPDC2010, (2010)

[5] Geoffrey Fox and Shrideep Pallickara. Deploying the NaradaBrokering Substrate in Aiding Efficient Web & Grid Service Interactions. Invited paper for Special Issue of the Proceedings of the IEEE on Grid Computing. Vol 93, No 3, 564-577(2005).