|  |  |
| --- | --- |
|  | **project 3**  Scalability of Big Data workloads in YARN Environment  **Submitted by:**  Hoda Moradi  (1001121261)  Padmavati H Channal  (1001398166)  We have neither given nor received any unauthorized assistance on this work  14 December 2016 |
|  |  |

**Contents**

|  |  |  |
| --- | --- | --- |
| **Serial No:** | **Topic** | **Page no** |
| 1. | Introduction | 3 |
| 2 | How did we solve the problems & What is our learning? | 4 |
| 3 | Workload selection and Configurations | 5 |
| 4 | Experimental Results | 7 |
| 5 | Possible scalability issues | 8 |
| 6 | References | 9 |

**1.Introduction**

**Scalability of Big Data workloads in a YARN environment: Why?**

**Classical Hadoop MapReduce:**

The large number of responsibilities given to a single process causes significant scalability issues, especially on a larger cluster where the JobTracker has to constantly keep track of thousands of TaskTrackers, hundreds of jobs, and tens of thousands of map and reduce tasks. On the contrary, the TaskTrackers usually run only a dozen or so tasks, which were assigned to them by the hard-working JobTracker

YARN is a completely rewritten architecture of Hadoop cluster. It seems to be a game-changer for the way distributed applications are implemented and executed on a cluster of commodity machines.

YARN offers clear advantages in scalability, efficiency, and flexibility compared to the classical MapReduce engine in the first version of Hadoop. Both small and large Hadoop clusters greatly benefit from YARN. To the end user (a developer, not an administrator), the changes are almost invisible because it's possible to run unmodified MapReduce jobs using the same MapReduce API and CLI.

***Assignments completed:***

* Assignment-1
* Assignment-2
* Assignment-3

***Virtual Machine Used:***

* Team: hoda\_padm
* Ip address: 10.255.255.150
* **VM: kvm5**
* **Password: 123456**

**2.How did we solve the problems and what we learnt?**

Problems we encountered:

-Setting up the YARN environment.

- Deploy the Intel Big Data benchmarks HiBench in the VM.

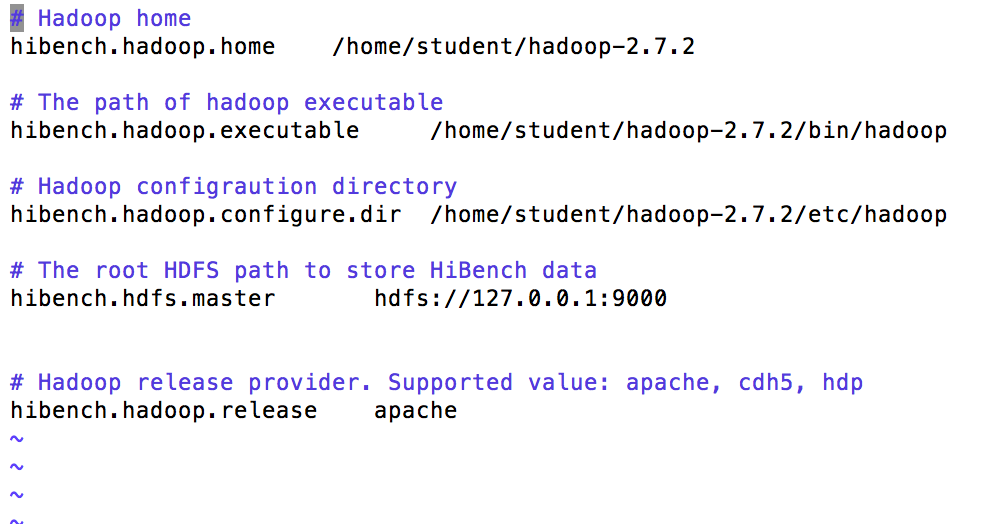
- Configuring Hadoop.conf

- Recognizing the right build for spark ( we build for 1.6 but we had 2.0.2)

How did we solve?

-We experimented with different values of master path configured in Hadoop.conf

Solution: Right Configuration



Other configurations include:

* Yarn.xml
* 99-User conf

Our Learning:

Right configurations of all the files are very important.

**3. Work load selection:**

**IntSum example, a Hadoop job run in a Yarn environment**

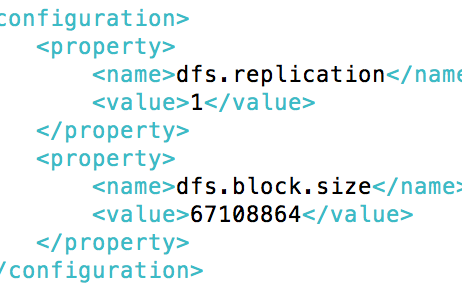
* “**IntSum**” takes in inputs as arithmetic numbers and calculates the sum from the given inputs.
* IntSum.jar file is run in the Hadoop-2.7.2 folder as follows:
  + bin/**hadoop** **jar** /home/student/**IntSum.jar** /input/**input**.**txt** /**result**

This example of IntSum which is run in Yarn environment is configured with many configuration files including hadoop.conf, yarn-site.xml and 99-users.conf

**Inputs to the workload:**

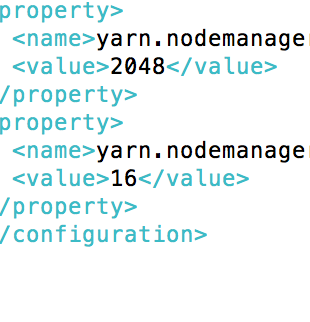
**# Mappers of the job:**

To change the number of mappers of the job we update the hdfs-site.xml “Block size” with different values.

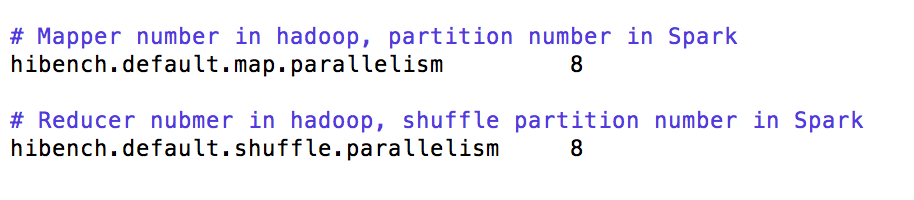


**# Containers on the worker node:**

To change the number of containers of the job we update the yarn-site.xml with different values.



Configurations in Hibench.conf:

****

**4 Experimental Results:**

**Table 1: Containers Vs Time spent**

|  |  |  |
| --- | --- | --- |
| resource.memory-mb | resource.cpu-vcores | **Time taken(ms)** |
| **1024** | **8** | **1680** |
| **2048** | **8** | **1740** |
| **4096** | **8** | **1820** |
| **1024** | **16** | **1700** |
| **2048** | **16** | **1710** |
| **4096** | **16** | **1760** |
| **1024** | **32** | **1640** |
| **2048** | **32** | **1670** |
| **4096** | **32** | **1640** |

**Job configuration: Block size- 64MB**

**Table 2: Block Size Vs Time spent**

**Job Configuration: Resource Memory: 2048, Resource CPU-Vcores**

|  |  |
| --- | --- |
| **Block size** | **Time spent (ms)** |
| **32 MB** | **1580** |
| **64 MB** | **1690** |
| **128 MB** | **1820** |

**Table 3: Reducers Vs Time spent**

|  |  |
| --- | --- |
| **Shuffle parallelism** | **Time spent (ms)** |
| **2** | **1730** |
| **4** | **1740** |
| **8** | **1670** |
| **16** | **1750** |

**5. Possible Scalable issues:**

**Explanation for Table1:**

**Conclusion:** Increase inresource.memory-mb increases the time spent

**Why:**

* Depends on how many data nodes do I have?
* Depends on how many disks and cores do we have on each node

**Explanation for Table2:**

**Conclusion:** Increase in Block size increases the time spent

**Why?**

* Small files are not recommended
* MapReduce tasks operate on one block at a time, so if having too few tasks (less then nodes in cluster), your jobs might be slow.

**Explanation for Table3:**

**Conclusion:** Increase in Reducers increases the time spent

**Why?**

* Each reduce need to start up & instantiated in the nodes, which result in an increase of startup time.
* Data need to be split across the entire number of reducers which require more network transfer time and parsing time.

**6. References:**

[**http://www.slideshare.net/vgogate/hadoop-configuration-performance-tuning**](http://www.slideshare.net/vgogate/hadoop-configuration-performance-tuning)

[**https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#YARN\_on\_Single\_Node**](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html#YARN_on_Single_Node)

[**https://github.com/intel-hadoop/HiBench/blob/master/README.md**](https://github.com/intel-hadoop/HiBench/blob/master/README.md)