Supporting a Hadoop Cluster with Unused PCs

Floriane Le Floch - Arthur Busser - Paul Dennetiere

December 17, 2016

Summary

Part I

Introduction

0.1 Personal computers

In 2008, approximately 1 billion personal computers were in use worldwide. Today, that number stands above 2 billion[1]. In the United States of America, Japan, Germany, the United Kingdom, France, and Italy, the number of PCs gets closer to the number of inhabitants every year[2].

In 2016, more than half (50.1%) of the world's population was an Internet user[3]. In this Information Age, most computers are connected to each other and can easily communicate. Countless companies rely on computers for communication, marketing, accounting, storage, documents and reports, education, and research.

As things stand, personal computers used for work and/or personal activities add up to an immense amount of processing power and that amount is growing continuously.

0.2 The problem

Very few people use the full computing capabilities of their PCs. Even then, there are long periods of time every day when PCs are left unused. In many companies, almost every employee is provided with a PC. At night, desktop computers - and laptops too - are often left on, staying idle until their user returns in the morning. During this time, those PCs could be doing something.

The main problem with personal computers is that they are personal. The average person will never use the full potential of their PC, simply because most of the time they will not be using it. This results in a huge amount of processing power going to waste. If only the computer's capabilities could be shared somehow...

0.3 Our idea

As we stated above, most computers are connected and can easily communicate with each other. Many professionals have used this to their advantage, organizing computers into clusters that could tackle computation problems as a whole.

Our idea is to create a tool that, once installed on a personal computer, would wait for the PC to be unused to connect it to a remote computer cluster. The cluster could then use the new node's processing power to advance in its current calculations. When the PC would once again need to be used, it would disconnect from the cluster and continue as it normally does.

[1]Source: Forrester Research
[2]Source: Computer Industry Almanac Inc.
[3]Source: http://www.internetworldstats.com/

Part II

Our solution

We want to make unused computers available for an Hadoop Cluster. In order to do that, we want to check at regular intervals if a computer is used or not (i.e. is in sleep mode). Our goal is to dynamically adding sleeping computer to the cluster, in order to make more ressources available. In order to achieve this, we will have to ensure several things. This part will deals with the general structure, and what this structure induces for the cluster and added computers.

Chapter 1

Setting up DataNodes on an unused computers

First we will have to make the computer we want to add able to communicate with the Master (NameNode). We have several options to achieve this :

- Virtualization
- Use only linux distribution
- Application containers

Only using linux distributions will greatly impact the interest of this project, so it's clearly not a solution. Virtualization doesn't seem good either, because of its need of ressources, and the general weight of those solutions. That's why we will use an Application Container : Docker.

1.1 Docker

Docker define itself as :

> "Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in."

Therefore by using Docker we will be able to use a linux application on every operating system encountered in the company. Moreover, by doing a snapshot of a running configuration, Docker allows us to deploy a solution very easily, and with a good velocity. Finally, the global strategy is to have a "Slave Image" for Docker : Every computer of the company will have Docker installed on it, and will start Docker application every

time the computer goes in sleep mode (we can easily do this through a C# script, running in background, checking the sleep mode, and launching Docker application, or even using TaskLauncher included in Microsoft's OS).

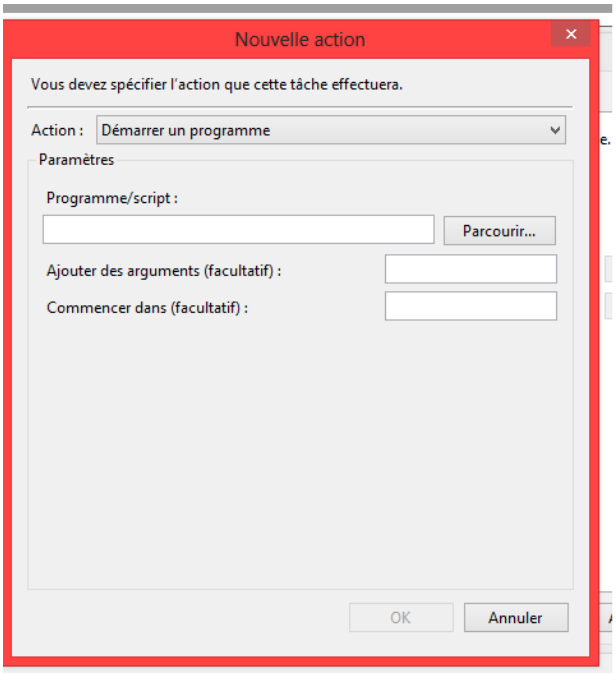

Figure 1.1: Screen of Microsoft's Task Launcher

1.2 Configuration

We supposed that a "classical Hadoop cluster" is already configured (meaning we do not install a full cluster). So our goal is to provide Hadoop slave service on the computer, and to make it communicate with the Master. We will first create an image of a running configuration. Such images are already available on Docker's Hub (Docker's image sharing website), but we will assume that we can't get such images : The first thing to do is to create a linux distributions image, a Debian for example. After that we install Hadoop slave service on this Debian.

- Creating Hadoop user and password :

```
useradd hadoop
passwd hadoop
```

- Install Slave service on all slave by running on Master :

```
# su hadoop
```

```
$ cd /opt/hadoop
$ scp -r hadoop hadoop-slave-1:/opt/hadoop
$ scp -r hadoop hadoop-slave-2:/opt/hadoop
```

- Setup Password less connectivity from master to new slave.

  ```
  mkdir -p $HOME/.ssh
  chmod 700 $HOME/.ssh
  ssh-keygen -t rsa -P '' -f $HOME/.ssh/id_rsa
  cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
  chmod 644 $HOME/.ssh/authorized_keys
  scp $HOME/.ssh/id_rsa.pub hadoop@IPPATH:/home/hadoop/
  ```

  Where IPPATH is the IP address of the new node. And make the same thing on the slaves, in order to exchange ssh keys between master and slaves.

- Then set a Hostname on new slaves in /etc/sysconfig/network :

  ```
  NETWORKING=yes
  HOSTNAME=slaveX.in
  ```

  Where X is the node Id (arbitrary setted, but unique)

- Run on every new slave the following, to make changes effective :

  ```
  hostname slaveX.in
  ```

- Update /etc/hosts on all machines of the cluster with the following lines:

  ```
  IPPATH slaveX.in slaveX
  ```

By using this configuration we are able to dynamically add a data node to an existing cluster. The only point of failure is on IP addresses : every node should have a different one, and use it in order to have a functionnal network, however, this IP address can be found on the host.

Finally, the new slave have to start HDFS by using this command :

```
./bin/hadoop-daemon.sh start datanode
```

This last command will have to be execute every time the docker application get started.

1.3 Conclusion

In conclusion, docker is a tool (application container) that will allow us to run hadoop on pretty much every system, and moreover allow us to have a configuration routine that pratically does not differ from a machine to another. By porting this solution on every

computer of the company, we can assume that every computers going in sleep mode, and having a docker image configured like above, will connect to the master, and be part of the Hadoop cluster.

However we can easily imagine that many difficulties will come up, and we will discuss them in the following chapter.

Part III

Difficulties that we expect to encounter

Chapter 2

Redundancy Problems

2.1 What we expect to happen

Because of bla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla