Cloud computing

Filip Gwardecki

January 23, 2021

1 Introduction

The main purpose of that project was to learn how to establish connection between at least two EC2 instances on Amazon Web Services. Cloud computing is getting more and more popular. It is one of best solution when the calculation require a lot of computing power, but are not used all the time. There is no need to maintain expensive machines, while they are used only for couple of hours.

2 Heat diffusion

EC2 instances were connected by Spark Apache in Standalone Mode. They were used to calculate a simple heat diffusion in one dimension,which is described by the equation:

$$\frac{\partial u}{\partial t} = \lambda \Delta u \tag{1}$$

Which in one dimension is reduced to:

$$\frac{\partial u}{\partial t} = \lambda \frac{\partial^2 u}{\partial x^2} \tag{2}$$

Now it can be easily discretized to:

$$u_i^{n+1} = u_i^n + \frac{\lambda \Delta t}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n) \tag{3}$$

where upper index stands for step in time and lower index stands for position (i-1 is left neighbour of i).

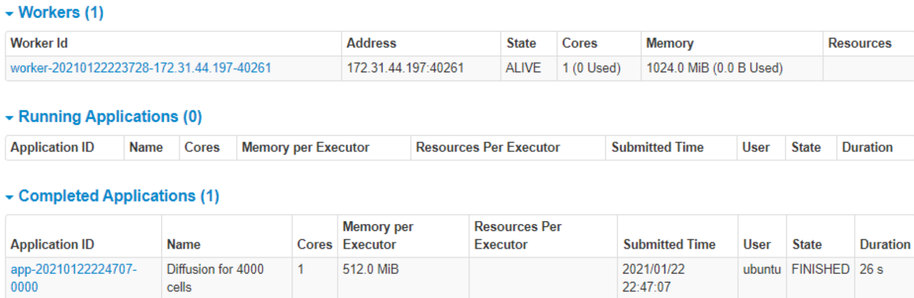
Code was parallized using PySpark and is listed below:

Listing 1: 1-D Diffusion

```python
import numpy
from pyspark import SparkContext
def Diffusion(ncells):
    sc = SparkContext("your spark URL here", f'Diffusion for {ncells} cells')
    leftX=-1000.
    rightX=+1000.
    sigma=300.
    ao=1.
    coeff=.375
    dx = (rightX-leftX)/(ncells-1)
    def tempFromIdx(i):
        x = leftX + dx*i + dx/2
        return (i, ao*numpy.exp(-x*x/(2.*sigma*sigma)))

    def interior(ix):
        return (ix[0] > 0) and (ix[0] < ncells-1)

    def stencil(item):
        i,t = item
        vals = [ (i,t) ]
        cvals = [ (i, -2*coeff*t), (i-1, coeff*t), (i+1, coeff*t) ]
        return vals + list(filter(interior, cvals))

    temp = map(tempFromIdx,range(ncells))
    data= sc.parallelize(temp)
    for i in range(50):
        stencilParts = data.flatMap(stencil)
        data = stencilParts.reduceByKey(lambda x,y:x+y)
    result = data.collect()
    sc.stop()

    return result
```

3 Results

While struggling with a lot of data to calculate it is hard to select how many machines should be used. Too few machines may cause some trouble with lack of memory. On the other hand, too many of them does not give satisfactory reduce in time comparing to optimum number of them.
I have tried to designate an optimum of used cores for size of calculation, which was counted as a number of steps. After some tries I set it for 4000 steps per 1 core. Time needed for calculation was approximately the same while scaling used cores.



Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210122223728-172.31.44.197-40261	172.31.44.197:40261	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

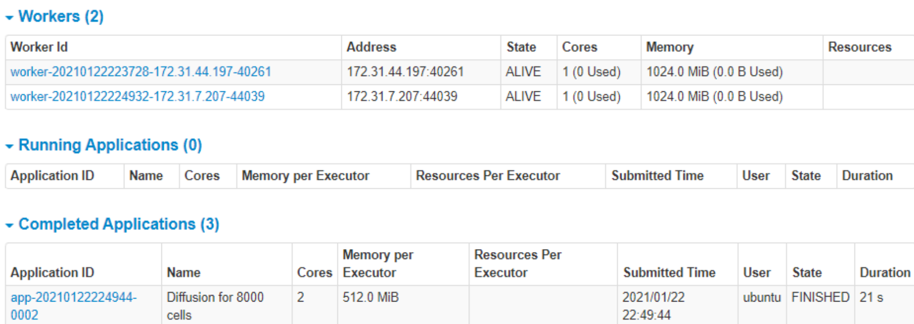
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210122224707-0000	Diffusion for 4000 cells	1	512.0 MiB		2021/01/22 22:47:07	ubuntu	FINISHED	26 s

Figure 1: 1 core - 4000 nodes



Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210122223728-172.31.44.197-40261	172.31.44.197:40261	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122224932-172.31.7.207-44039	172.31.7.207:44039	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

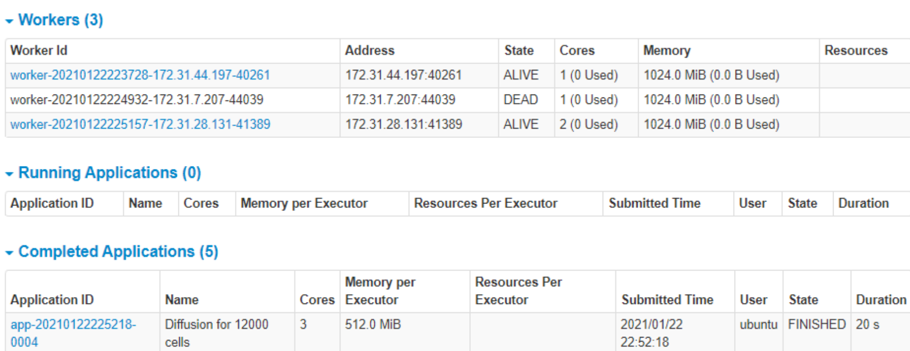
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210122224944-0002	Diffusion for 8000 cells	2	512.0 MiB		2021/01/22 22:49:44	ubuntu	FINISHED	21 s

Figure 2: 2 cores - 8000 nodes



Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210122223728-172.31.44.197-40261	172.31.44.197:40261	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122224932-172.31.7.207-44039	172.31.7.207:44039	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225157-172.31.28.131-41389	172.31.28.131:41389	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (5)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210122225218-0004	Diffusion for 12000 cells	3	512.0 MiB		2021/01/22 22:52:18	ubuntu	FINISHED	20 s

Figure 3: 3 cores - 12000 nodes

Workers (4)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210122223728-172.31.44.197-40261	172.31.44.197:40261	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122224932-172.31.7.207-44039	172.31.7.207:44039	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225157-172.31.28.131-41389	172.31.28.131:41389	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225338-172.31.25.33-33401	172.31.25.33:33401	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (7)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210122225402-0006	Diffusion for 16000 cells	4	512.0 MiB		2021/01/22 22:54:02	ubuntu	FINISHED	21 s

Figure 4: 4 cores - 16000 nodes

Workers (5)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210122223728-172.31.44.197-40261	172.31.44.197:40261	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122224932-172.31.7.207-44039	172.31.7.207:44039	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225157-172.31.28.131-41389	172.31.28.131:41389	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225338-172.31.25.33-33401	172.31.25.33:33401	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225526-172.31.7.207-44825	172.31.7.207:44825	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

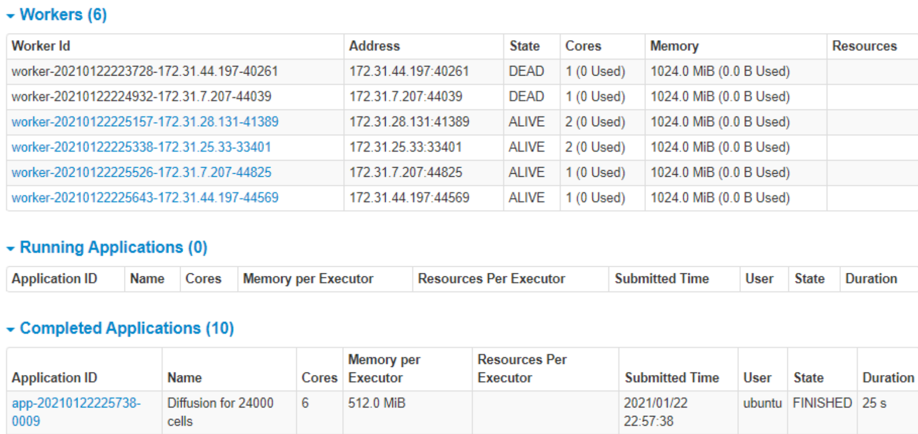
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (8)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210122225527-0007	Diffusion for 20000 cells	5	512.0 MiB		2021/01/22 22:55:27	ubuntu	FINISHED	22 s

Figure 5: 5 cores - 20000 nodes



Workers (6)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210122223728-172.31.44.197-40261	172.31.44.197:40261	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122224932-172.31.7.207-44039	172.31.7.207:44039	DEAD	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225157-172.31.28.131-41389	172.31.28.131:41389	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225338-172.31.25.33-33401	172.31.25.33:33401	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225526-172.31.7.207-44825	172.31.7.207:44825	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
worker-20210122225643-172.31.44.197-44569	172.31.44.197:44569	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (10)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210122225738-0009	Diffusion for 24000 cells	6	512.0 MiB		2021/01/22 22:57:38	ubuntu	FINISHED	25 s

Figure 6: 6 cores - 24000 nodes

References

[1] Discretizing 1-D diffusion:
`https://github.com/barbagroup/CFDPython/blob/master/lessons/04_Step_3.ipynb`

[2] 1-D Diffusion in PySpark:
`https://www.dursi.ca/post/hpc-is-dying-and-mpi-is-killing-it.html`