UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

INF-464 Computación Distribuida para Big Data
Segundo Semestre 2021

Apache Spark:
AWS EMR vs Local Host

Héctor Labraña Rojas hector.labrana.13@sansano.usm.cl

Miércoles 15 de Diciembre del 2021

❖ Contenidos

❖ Motivación

1. Pyspark

1. Framework Open Source.
2. Computación en paralelo.
3. Procesamiento de gran cantidad de datos y/o modelos complejos (Big Data)
4. Fácil de instalar y utilizar.

2. Costos AWS

1 instance(s) x 0.048 USD hourly x (100 / 100 Utilized/Month) x 730 hours in a month = 35.0400 USD (EMR master node cost)

EMR master node cost (monthly): 35.04 USD

2 instance(s) x 0.048 USD hourly x (100 / 100 Utilized/Month) x 730 hours in a month = 70.0800 USD (EMR task node cost)

EMR task node cost (monthly): 70.08 USD

Amazon EMR estimate

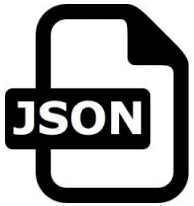
Total monthly cost:	**105.12 USD**

https://databricks.com/wp-content/uploads/2018/12/PySpark-1024x164.png
https://calculator.aws/#/createCalculator/EMR

❖ Motivación

3. Contenido Dataset

RC_2013-01.bz2	BZIP2 Compressed Reddit Comments (JSON objects)	2,830,544,500	Sep 9 2016 2:01 PM



RC_2013-01.json

17,43 [GB]

```
|-- archived: boolean (nullable = true)
|-- author: string (nullable = true)
|-- author_flair_css_class: string (nullable = true)
|-- author_flair_text: string (nullable = true)
|-- body: string (nullable = true)
|-- controversiality: long (nullable = true)
|-- created_utc: string (nullable = true)
|-- distinguished: string (nullable = true)
|-- downs: long (nullable = true)
|-- edited: string (nullable = true)
|-- gilded: long (nullable = true)
|-- id: string (nullable = true)
|-- link_id: string (nullable = true)
|-- name: string (nullable = true)
|-- parent_id: string (nullable = true)
|-- removal_reason: string (nullable = true)
|-- retrieved_on: long (nullable = true)
|-- score: long (nullable = true)
|-- score_hidden: boolean (nullable = true)
|-- subreddit: string (nullable = true)
|-- subreddit_id: string (nullable = true)
|-- ups: long (nullable = true)
```

http://files.pushshift.io/reddit/comments/

❖ Motivación

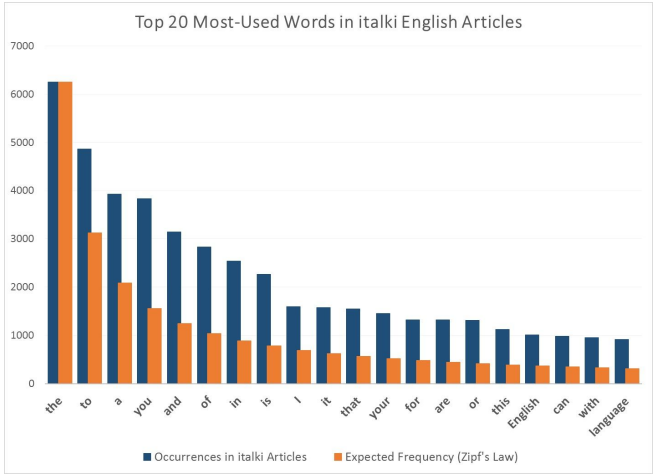
4. Ley de Zipf

reddit

JSON

RC_2013-01.json

17,43 [GB]



https://blogging-techies.com/wp-content/uploads/2020/06/1593316919_La-ley-de-Zipf-sobre-el-idioma-ingles-te-volvera.jpg
http://image.slidesharecdn.com/ndices-de-texto-1206833828888091-5/95/ndices-de-texto-5-728.jpg

❖ Objetivo

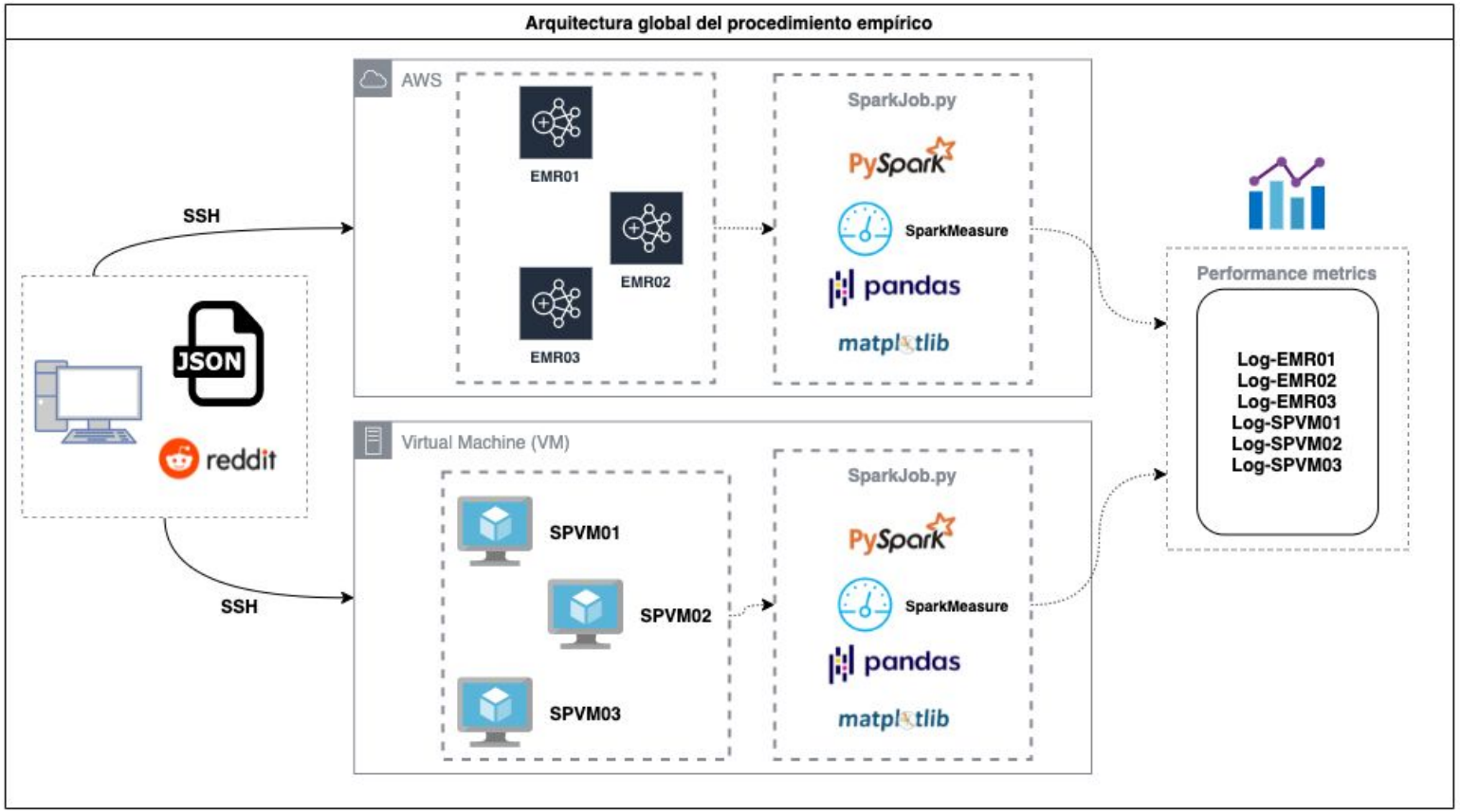
1. Objetivo Principal

> Obtener el costo de oportunidad existente entre utilizar AWS EMR versus el uso de un servidor local (localhost) bajo un contexto de rendimiento y procesamiento en batch vía Pyspark.

2. Objetivos Específicos

1. Crear y configurar un clúster de AWS EMR.
2. Diseñar un flujo de procesamiento de datos para tecnologías de Big Data.
3. Comprobar la Ley de Zipf (Palabras frecuentes)
4. Ejecución de procesos vía SSH.
5. Instalación de software y entorno Big Data.

❖ Diseño Arquitectura



❖ Diseño de Pruebas - Configuración

EMR

```
H/W path     Device          Class          Description
=====================================================
                             system         m5.xlarge
/0                           bus            Motherboard
/0/0                         memory         64KiB BIOS
/0/4                         processor      Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz
/0/4/5                       memory         1536KiB L1 cache
/0/4/6                       memory         24MiB L2 cache
/0/4/7                       memory         35MiB L3 cache
/0/8                         memory         16GiB System Memory
/0/8/0                       memory         16GiB DIMM DDR4 Static column Pseudo-static Syr
/0/100                       bridge         440FX - 82441FX PMC [Natoma]
/0/100/1                     bridge         82371SB PIIX3 ISA [Natoma/Triton II]
/0/100/1.3                   generic        82371AB/EB/MB PIIX4 ACPI
/0/100/3                     display        Amazon.com, Inc.
/0/100/4                     storage        Amazon.com, Inc.
/0/100/5     eth0            network        Elastic Network Adapter (ENA)
/0/100/1e                    storage        Amazon.com, Inc.
/0/100/1f                    storage        Amazon.com, Inc.
/0/1                         system         PnP device PNP0b00
/0/2                         input          PnP device PNP0303
/0/3                         input          PnP device PNP0f13
/0/5                         printer        PnP device PNP0400
/0/6                         communication  PnP device PNP0501
/1           veth2897ea8     network        Ethernet interface
/2           docker0         network        Ethernet interface
/3           veth3db5e7b     network        Ethernet interface
/4           br-419132092e1d network        Ethernet interface
```

Local Host (SPVM)

```
H/W path      Device      Class          Description
=========================================================
                          system         Standard PC (i440FX + PIIX, 1996)
/0                        bus            Motherboard
/0/0                      memory         96KiB BIOS
/0/400                    processor      Intel Xeon E312xx (Sandy Bridge, IBRS update)
/0/401                    processor      Intel Xeon E312xx (Sandy Bridge, IBRS update)
/0/402                    processor      Intel Xeon E312xx (Sandy Bridge, IBRS update)
/0/403                    processor      Intel Xeon E312xx (Sandy Bridge, IBRS update)
/0/1000                   memory         16GiB System Memory
/0/1000/0                 memory         16GiB DIMM RAM
/0/100                    bridge         440FX - 82441FX PMC [Natoma]
/0/100/1                  bridge         82371SB PIIX3 ISA [Natoma/Triton II]
/0/100/1.1                storage        82371SB PIIX3 IDE [Natoma/Triton II]
/0/100/1.3                bridge         82371AB/EB/MB PIIX4 ACPI
/0/100/2                  display        QXL paravirtual graphic card
/0/100/3                  network        Virtio network device
/0/100/3/0    ens3        network        Ethernet interface
/0/100/4                  multimedia     82801FB/FBM/FR/FW/FRW (ICH6 Family) High Definition Audio Controller
/0/100/5                  bus            82801I (ICH9 Family) USB UHCI Controller #1
/0/100/5/1    usb2        bus            UHCI Host Controller
/0/100/5.1                bus            82801I (ICH9 Family) USB UHCI Controller #2
/0/100/5.1/1  usb3        bus            UHCI Host Controller
/0/100/5.2                bus            82801I (ICH9 Family) USB UHCI Controller #3
/0/100/5.2/1  usb4        bus            UHCI Host Controller
/0/100/5.7                bus            82801I (ICH9 Family) USB2 EHCI Controller #1
/0/100/5.7/1  usb1        bus            EHCI Host Controller
/0/100/6                  communication  Virtio console
/0/100/6/0                generic        Virtual I/O device
/0/100/7                  storage        Virtio block device
/0/100/7/0    /dev/vda    disk           68GB Virtual I/O device
/0/100/7/0/1  /dev/vda1   volume         1023KiB BIOS Boot partition
/0/100/7/0/2  /dev/vda2   volume         63GiB EXT4 volume
/0/100/8                  generic        Virtio memory balloon
/0/100/8/0                generic        Virtual I/O device
/0/1                      system         PnP device PNP0b00
/0/2                      input          PnP device PNP0303
/0/3                      input          PnP device PNP0f13
/0/4                      storage        PnP device PNP0700
/0/5                      communication  PnP device PNP0501
/0/6          scsi0       storage
/0/6/0.0.0    /dev/cdrom  disk           QEMU DVD-ROM
```

> sudo lshw -short

❖ Diseño de Pruebas - Ejecución

SSH

```
> PYTHONSTARTUP=SparkJob.py pyspark
```

```python
# Instalando Dependencias
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode,split,col
from sparkmeasure import StageMetrics

spark = SparkSession.builder.getOrCreate()
stagemetrics = StageMetrics(spark)

PATH = './data/RC_2013-01'

stagemetrics.begin()
df = spark.read.json(PATH)

df.printSchema()
df.count()

comments = df.select(df.body)
comments.show()

SplitData = comments.withColumn('body',explode(split("body"," ")))

frequentWords = SplitData.groupby('body').count().orderBy(col('count').desc())
frequentWords.show()
frequentWords.count()

frequentWords = frequentWords.filter(frequentWords.body != "")
frequentWords.show()
frequentWords.count()

Top15 = spark.createDataFrame(frequentWords.take(15)).toPandas()
Top15

stagemetrics.end()
stagemetrics.print_report()

# Generar gráfica
import matplotlib.pyplot as plt

plt.plot(Top15['body'],Top15['count'],color='red')
plt.bar(Top15['body'],Top15['count'],color='cyan')
plt.xlabel('Top 10 Most Frequent Words')
plt.ylabel('Absolute Frequency')
plt.savefig('plotResults.png')
```

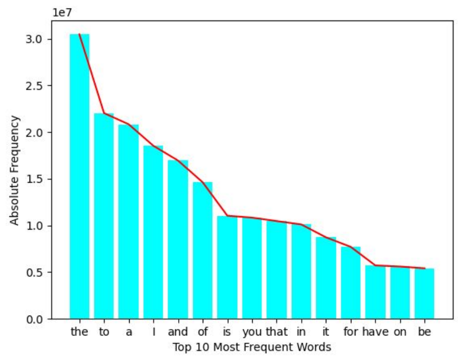
```
Scheduling mode = FIFO
Spark Context default degree of parallelism = 4
Aggregated Spark stage metrics:
numStages => 17
numTasks => 1086
elapsedTime => 1809344 (30 min)
stageDuration => 1806240 (30 min)
executorRunTime => 7175023 (2.0 h)
executorCpuTime => 5950430 (1.7 h)
executorDeserializeTime => 14380 (14 s)
executorDeserializeCpuTime => 5207 (5 s)
resultSerializationTime => 80 (80 ms)
jvmGCTime => 58597 (59 s)
shuffleFetchWaitTime => 7 (7 ms)
shuffleWriteTime => 7708 (8 s)
resultSize => 618323 (603.0 KB)
diskBytesSpilled => 0 (0 Bytes)
memoryBytesSpilled => 0 (0 Bytes)
peakExecutionMemory => 20719861760
recordsRead => 212561090
bytesRead => 122094986351 (113.0 GB)
recordsWritten => 0
bytesWritten => 0 (0 Bytes)
shuffleRecordsRead => 263863495
shuffleTotalBlocksFetched => 22507
shuffleLocalBlocksFetched => 11016
shuffleRemoteBlocksFetched => 11491
shuffleTotalBytesRead => 5207503080 (4.0 GB)
shuffleLocalBytesRead => 2603662194 (2.0 GB)
shuffleRemoteBytesRead => 2603840886 (2.0 GB)
shuffleRemoteBytesReadToDisk => 0 (0 Bytes)
shuffleBytesWritten => 5207503080 (4.0 GB)
shuffleRecordsWritten => 263863495
>>>
```

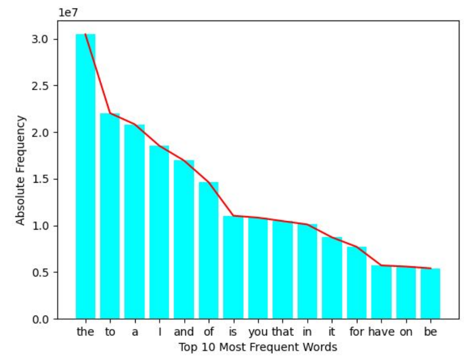
❖ Resultados - EMR

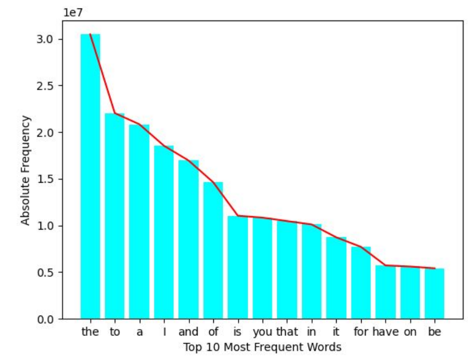
Métrica	EMR01	EMR02	EMR03	EMR
Scheduling mode	FIFO	FIFO	FIFO	FIFO
numStages	17	17	17	17
numTasks	1086	1086	1086	1086
elapsedTime [ms]	1762201	1809344	1759065	1776870
elapsedTime [min]	29	30	29	29
executorDeserializeTime [s]	16	14	11	14
executorDeserializeCpuTime [s]	5	5	5	5
resultSerializationTime [ms]	133	80	88	100
jvmGCTime [min]	1	1	1	1
shuffleFetchWaitTime [ms]	11	7	14	11
shuffleWriteTime [min]	0,12	0,13	0,12	0,12
resultSize [KB]	603	603	603	603
peakExecutionMemory [MB]	19760	19760	19760	19760
recordsRead	212561090	212561090	212561090	212561090
bytesRead [GB]	113	113	113	113
shuffleTotalBytesRead [GB]	4	4	4	4
shuffleBytesWritten [GB]	4	4	4	4
shuffleRecordsWritten	263863495	263863495	263863495	263863495

❖ Resultados - EMR

EMR01	EMR02	EMR03





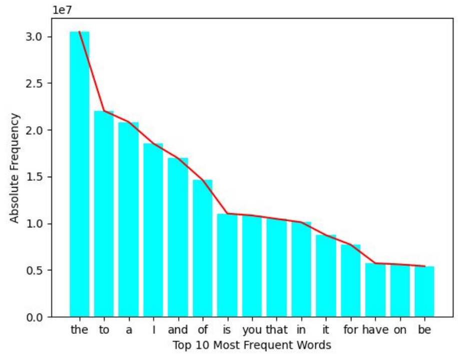


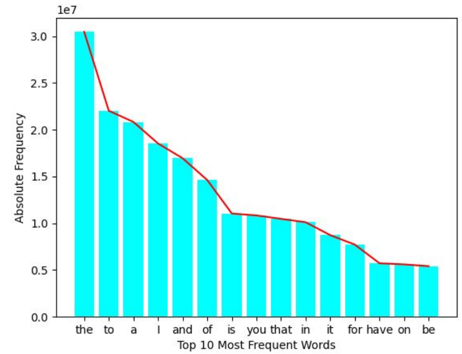
❖ Resultados - Localhost (SPVM)

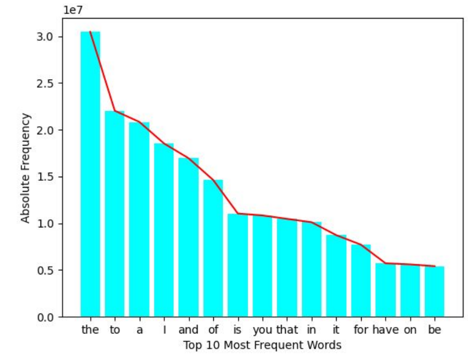
Métrica	SPVM01	SPVM02	SPVM03	Localhost
Scheduling mode	FIFO	FIFO	FIFO	FIFO
numStages	17	17	17	17
numTasks	1918	1918	1918	1918
elapsedTime [ms]	1904033	2582716	1939427	2142059
elapsedTime [min]	32	43	32	36
executorDeserializeTime [s]	11	14	12	12
executorDeserializeCpuTime [s]	9	11	9	10
resultSerializationTime [ms]	78	141	77	99
jvmGCTime [min]	2,4	4,8	2,7	3,3
shuffleFetchWaitTime [ms]	7	48	70	42
shuffleWriteTime [min]	1,2	2,7	1,2	1,7
resultSize [KB]	1364	1364	1364	1364
peakExecutionMemory [MB]	5060	5060	5060	5060
recordsRead	212561090	212561090	212561090	212561090
bytesRead [GB]	113	113	113	113
shuffleTotalBytesRead [GB]	4	4	4	4
shuffleBytesWritten [GB]	4	4	4	4
shuffleRecordsWritten	263863835	263863835	263863835	263863835

❖ Resultados - Localhost (SPVM)

SPVM01	SPVM02	SPVM03







❖ Resultados - Comparativo

Métrica	EMR	Localhost	EMR-Localhost	Error [%]
Scheduling mode	FIFO	FIFO	FIFO	0,0
numStages	17	17	0	0,0
numTasks	1086	1918	-832	43,4
elapsedTime [ms]	1776870	2142059	-365189	17,0
elapsedTime [min]	29	36	-6	17,8
executorDeserializeTime [s]	14	12	1	10,8
executorDeserializeCpuTime [s]	5	10	-5	48,3
resultSerializationTime [ms]	100	99	2	1,7
jvmGCTime [min]	1	3,3	-2,3	69,7
shuffleFetchWaitTime [ms]	11	42	-31	74,4
shuffleWriteTime [min]	0,12	1,7	-1,58	92,7
resultSize [KB]	603	1364	-761	55,8
peakExecutionMemory [MB]	19760	5060	14700	290,5
recordsRead	212561090	212561090	0	0,0
bytesRead [GB]	113	113	0	0,0
shuffleTotalBytesRead [GB]	4	4	0	0,0
shuffleBytesWritten [GB]	4	4	0	0,0
shuffleRecordsWritten	263863495	263863835	-340	0,0

Media	65,6
Mediana	48,3

❖ Conclusiones - Directas

1. Se comprueba la ley de Zipf al obtener resultados coherentes con la teoría y anticipados en la sección de motivación.
2. Los resultados obtenidos para el dataset RC_2013-01 son equivalentes en cada simulación realizada, independientemente del entorno de ejecución.
3. El uso de memoria principal, el tiempo de escritura y tiempo de extracción de registros son la mayor diferencia obtenida entre ambos entornos, alcanzando casi 15 [GB] de diferencia en uso de memoria RAM por parte de AWS EMR.
4. Los procesos de lectura y deserialización no afectan en la diferencia de rendimiento entre ambos entornos.
5. En promedio, AWS EMR realiza 832 tareas menos, correspondiente a un 43,4% de diferencia con respecto a Localhost
6. La mediana de las métricas obtenidas alcanza un 48,3% de diferencia a favor de AWS EMR.
7. La media de las métricas obtenidas alcanza un 65,6% de diferencia a favor de AWS EMR.
8. AWS EMR es 6 minutos más rápido en procesar el dataset en estudio en comparación a Localhost, lo que deriva en una diferencia del 17,8%.

❖ Conclusiones - Indirectas

1. El uso de pyspark para el diseño de flujos de procesamiento de grandes datasets es **relevante**, debido a la imposibilidad de cargar estos completamente en memoria principal.
2. Las métricas obtenidas con mayor porcentaje de error indican que **la optimización en el uso de memoria principal** es la clave para comprender la mejora en rendimiento de AWS EMR con respecto al entorno Localhost.
3. Considerando que pueden haber estado en ejecución procesos externos al experimento, **no se puede dar un porcentaje aproximado** de mejora en rendimiento, obligando a determinar un rango de aceptación.
4. En la teoría, la media aritmética es sensible a datos atípicos, sin embargo, en este caso el valor obtenido puede considerarse como una **cota superior en la diferencia de rendimiento** de los sistemas en estudio.
5. Considerando los resultados obtenidos y las conclusiones previas, se tienen las condiciones de poder decir que AWS EMR obtiene un **mejor rendimiento** que un entorno Localhost en un **rango de aceptación promedio aproximado al [48,3%, 65,6%]**, lo que se refleja en un tiempo de ejecución medio de 6 minutos de diferencia.
6. No existen las condiciones de poder asociar una **relación lineal** entre tiempo de ejecución y tamaño del dataset.

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO
DE INFORMÁTICA

INF-464 Computación Distribuida para Big Data
Segundo Semestre 2021

Apache Spark:
AWS EMR vs Local Host

Héctor Labraña Rojas hector.labrana.13@sansano.usm.cl

Miércoles 15 de Diciembre del 2021