Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Herramientas de Software para Big Data.
Obligatorio

Métricas para la detección y predicción de fallas en discos duros basado en Big Data

Matías Séttimo - 152946
Docente: Pedro Bonillo
2020

Contents

Introducción:

La datalización no es una moda, sino el nuevo paradigma empresarial. Los líderes de todos los sectores e industrias son conscientes del alto valor que tienen los datos como activo estratégico para mejorar, ya sea las decisiones de sus negocios, utilizarlos como bienes vendibles o simplemente basar su negocio en el procesamiento y análisis de los datos.
Si bien en el mundo digital estos datos no son más que una colección de ceros y unos, estas colecciones se deben persistir en el tiempo para que luego puedan ser accedidas cuando se necesiten. Este requerimiento puede ir desde unos pocos megabytes, hasta miles de petabytes de información.
Actualmente estos datos se almacenan en grandes centros de datos que poseen una infinidad de unidades de almacenamiento, ya sea en forma discos magnéticos o unidades flash.
Aún hoy, dado el costo actual por GB de las unidades de almacenamiento, la mayoría de estos centros de datos todavía utilizan discos magnéticos. Para el funcionamiento de estos discos, el acceso a los datos se realiza mediante la lectura de una superficie magnetizada y para ello se requiere que partes mecánicas lean una superficie que gira a muy alta velocidad.
Estas operaciones mecánicas de lectura y escritura generan vibraciones, mucho calor, ruidos de alta frecuencia, e interferencias eléctricas que van decrementando la vida útil de las unidades.

Para cualquier centro de datos, el poder hacer un análisis de la vida útil de estas unidades, a fin de poder gestionar eficientemente los recursos, evitando las pérdidas de datos valiosos, mientras que se optimizan los costos, es crucial para llevar a cabo un negocio exitoso.

La idea de este documento es la de mostrar cómo podemos utilizar las herramientas vistas en clase para la gestión de grandes volúmenes de datos. A su vez, utilizarlas para poder hacer análisis y predicciones sobre un dataset que contiene el estado de salud diario de todas de las unidades de almacenamiento de un centro de datos.

Planteamiento del Problema

El problema a resolver es el de cómo podemos hacer para analizar un dataset que contiene millones de registros con los datos del estado de salud de una colección de discos duros. A fin de obtener información útil que pudiese ser utilizada para predecir cómo y cuándo fallarán las unidades de disco y cuáles son los fabricantes que construyen las unidades más robustas.
El dataset que utilizaremos contiene una instantánea diaria de los datos de salud de cada unidad de disco duro que está activa en el centro de datos.

Objetivo General

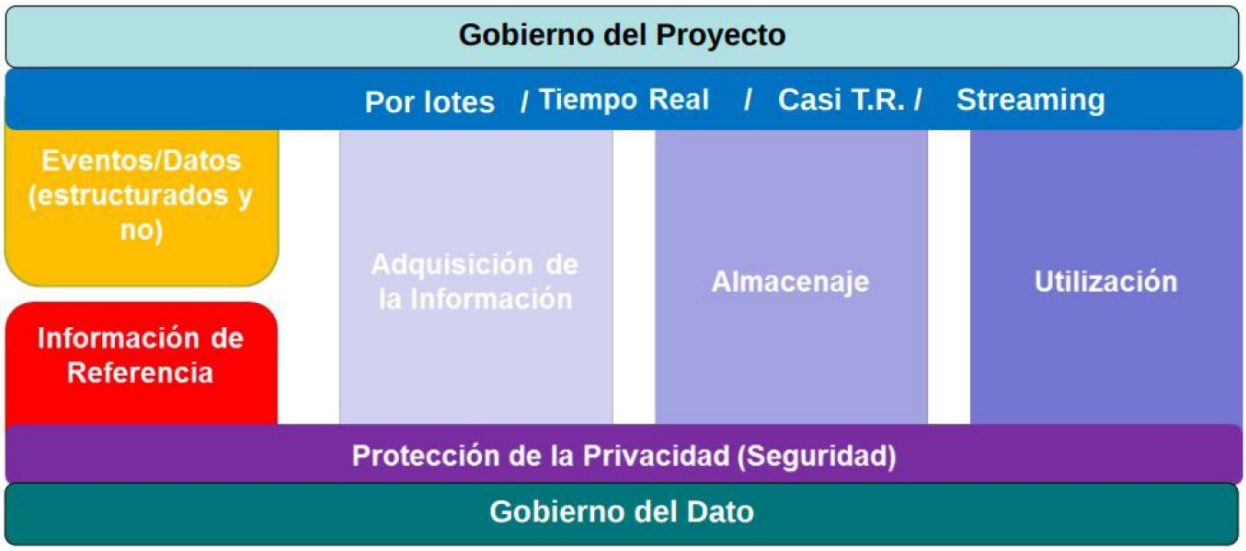
El objetivo es utilizar la fuente de datos sobre el estado de salud de una colección de discos duros de un cetro de datos y así poder obtener insights que permitan tomar decisiones estratégicas para hacer su reemplazo de forma eficiente y predecir sus fallas antes de que ocurran.

Objetivos Específicos

1. Analizar la fuente de datos.
2. Realizar una ingeniería de los atributos e ingesta de estos datos.
3. Indexar los datos para poder acceder a esto de forma rápida.
4. Generar vistas, reportes y predicciones en base a estos datos utilizando SQL de PySpark, Pandas y Machine Learning para generar predicciones.

Arquitectura de la solución

Para desarrollar este proyecto, me basé en una arquitectura de *Big Data* propuesta por el docente del curso



Dada esta arquitectura, esta entrega se concentró especialmente en el nivel técnico (Almacenaje y Utilización), ya que se trata de un proyecto pequeño para una materia de la universidad.

Descripción de las capas

- Fuentes de datos

En esta capa se establecen cuáles son las fuentes de datos con los que se va a trabajar. Se compone de datos, que pueden ser estructurados o desestructurados, o referencias a otros sistemas.

- Adquisición de la información:

Esta es la capa de la arquitectura donde se obtienen los datos de las fuentes definidas en la capa anterior (Fuentes de datos). Esta capa establece con qué herramientas se obtendrán y que tipo de ingesta realizarán: Baches, Micro baches, Tiempo real, etc.

- Almacenaje

Establece que medio se utilizará para almacenar los datos, para Big Data se utilizan, en general, herramientas para manejo de sistemas de archivos distribuidos, como HDFS de Hadoop. Luego se establece de qué forma se almacenarán los datos para su posterior procesamiento y análisis, se pueden utilizar Data Lakes, Data Warehouses o ambos. En esta capa también se define que procesamiento, destilería e indexación se hará sobre estos datos.

- Utilización:

Esta capa contiene la información procesada que pueden analizar los usuarios. Principalmente, estos datos pueden ser consultados directamente (Insights Tier), realizando consultas en Apache Hive, por ejemplo.
También existe otro componente que permite a los usuarios realizar distintos reportes sobre los datos con muchas dimensiones y profundidades.

Herramientas disponibles

Herramientas que utilizaré.

Justificación de las herramientas utilizadas.

1. Source o fuente de datos

La fuente de datos utilizada partió de los datos públicamente disponibles en el sitio web [1]del servicio de cloud storage Backblaze.
Esta empresa ofrece su dataset con el estado de salud de discos duros para cada trimestre del año desde el año 2013.
Backblaze ofrece estos datos con la condición de que se cite la fuente y que no se comercialicen estos datos ni los resultados o insights obtenidos a partir de estos.

[1] https://www.backblaze.com/b2/hard-drive-test-data.html

2. Adquisición de los datos.

La ingestión de los datos fue de tipo batch.
Todos los datos que se procesaron fueron descargados en archivos csv que fueron capturados de antemano por la fuente de datos mediante consultas a los parámetros S.M.A.R.T. de cada uno de los más de 150.000 discos del Datacenter.

Previo a la ingesta de estos datos se utilizo primero un script de Python que selecciona que columnas de la data estructurada se van a utilizar con el fin de reducir el tamaño del dataset. Luego se procedió a su lectura masiva mediante el uso del framework de Apache Spark (PySpark) en una jupyter notebook. Con un pyspark.sqlContext.

```
[6]: df_fullSet = sqlContext.read.options(header='True', delimiter=',') \
            .schema(schema) \
            .csv("../bigdata/**")
```

3. Almacenamiento y procesamiento.

Una vez cargados los datos en dataframes en memoria, estos fueron persistidos en formato Parquet columnar comprimido a fin de mejorar la performance de las consultas de los datos, además de aumentar la eficiencia del almacenamiento y procesamiento.

Para el procesamiento, se utilizó Apache Spark, con dataframes y funcionalidad de SQL ofrecida por SparkSQL para analizar la data a procesar.

Al utilizar Spark dentro de una jupyter notebook. Estas permitieron que no solo código ejecutable pueda mostrarse, sino que también se pueda visualizar y persistir los resultados de este código ejecutable para una posterior revisión.

4. Utilización.

Una vez procesados los datos por la capa anterior, estos se mostraron dentro del notebook de Jupyter para permitir hace un análisis o inspección visual del mismo. A su vez, se utilizaron mecanismos de machine learning para poder hacer regresiones lineales de los datos a fin de poder predecir por ejemplo la cantidad de discos duros que fallarían en el año 2021 como también el crecimiento en cantidad de unidades para el año 2021 y 2022.

La librería de **Pandas** permitió además que se pudieran graficar algunas tendencias, como por ejemplo fallas vs cantidad de discos en el Datacenter. El crecimiento en TBs del Datacenter. Etc.

Descripción del trabajo realizado

1. Fuente de datos.

La fuente de datos utilizada se obtuvo de los datos públicamente disponibles por el Datacenter Backblaze[2]. El cual deja por cada trimestre del año los datos crudos del estado salud de los discos duros, a la vez que ofrece en su blog el análisis de estos.
Estos datos crudos descargados van desde el primer trimestre del año 2013 hasta el 3er trimestre del año 2020.
Los datos se presentan en archivos .csv, uno por cada día del año, en un total de 2732 archivos cuyo tamaño total supera los 53GBs.
El dataset consta de múltiples columnas referidas a cada uno de los atributos S.M.A.R.T.[3] de los discos del Datacenter.

Aquí un extracto de los datos de cada tupla obtenida de Backblaze.

- **Date** – The date of the file in yyyy-mm-dd format.
- **Serial Number** – The manufacturer-assigned serial number of the drive.
- **Model** – The manufacturer-assigned model number of the drive.
- **Capacity** – The drive capacity in bytes.
- **Failure** – Contains a “0” if the drive is OK. Contains a “1” if this is the last day the drive was operational before failing.
- 2013-2014 SMART Stats – 80 columns of data, that are the Raw and Normalized values for 40 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- 2015-2017 SMART Stats – 90 columns of data, that are the Raw and Normalized values for 45 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- 2018 (Q1) SMART Stats – 100 columns of data, that are the Raw and Normalized values for 50 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- 2018 (Q2) SMART Stats – 104 columns of data, that are the Raw and Normalized values for 52 different SMART stats as reported by the given drive. Each value is the number reported by the drive.
- 2018 (Q4) SMART Stats – 124 columns of data, that are the Raw and Normalized values for 62 different SMART stats as reported by the given drive. Each value is the number reported by the drive.

Para nuestro análisis plenamente educativo/práctico. Se dejaron de lado los atributos SMART y solo se utilizarán las siguientes columnas.

```
|-- Date: date (yyyy-mm-dd)
|-- Serial_Number: string
|-- Model: string
|-- capacity_bytes: double
|-- Failure: integer
```

[2] https://www.backblaze.com/b2/hard-drive-test-data.html#downloading-the-raw-hard-drive-test-data
[3] https://es.wikipedia.org/wiki/S.M.A.R.T.

2. Ingeniería de los atributos e ingesta de datos.

Como se mencionó en la sección anterior. Se descartaron múltiples columnas del dataset a fin de reducir el tamaño del set de datos a utilizar para poder acelerar los procesos de consulta.

Para ellos se utilizó primeramente un script en Python que removió las columnas no utilizadas. Y solo conservó las mencionadas anteriormente.
El script *"procesar.py"*[4] *redujo el tamaño del dataset de 53.8 GB a 10.4 GB.*

```python
def processFile(path):
    print(path)
    f=pd.read_csv(path)
    keep_col = ['date','serial_number','model','capacity_bytes','failure']
    new_f = f[keep_col]
    new_f.to_csv(path, index=False)



def getListOfFiles(dirName):
    # create a list of file and sub directories
    # names in the given directory
    listOfFile = os.listdir(dirName)
    allFiles = list()
    # Iterate over all the entries
    for entry in listOfFile:
        # Create full path
        fullPath = os.path.join(dirName, entry)
        # If entry is a directory then get the list of files in this directory
        if os.path.isdir(fullPath):
            allFiles = allFiles + getListOfFiles(fullPath)
        else:
            allFiles.append(fullPath)

    return allFiles
```

Una vez reducido el tamaño del dataset se procedió a la ingesta de estos datos con apache Spark y el formato Apache Parquet[5].

Primero se definió el esquema de datos que se utilizará. La siguiente captura muestra como fue definido este esquema.

Date – The date of the file in yyyy-mm-dd format.
Serial Number – The manufacturer-assigned serial number of the drive.
Model – The manufacturer-assigned model number of the drive.
Capacity – The drive capacity in bytes.
Failure – Contains a "0" if the drive is OK. Contains a "1" if this is the last day the drive was operational before failing.

```python
schema = StructType() \
      .add("Date", DateType(),True) \
      .add("Serial_Number",StringType(),True) \
      .add("Model",StringType(),True) \
      .add("capacity_bytes",DoubleType(),True) \
      .add("Failure",IntegerType(),True)
```

[4] procesar.py está disponible dentro de los archivos de esta entrega.
[5] https://parquet.apache.org/

Una vez definido el esquema, se procedió a la ingesta de datos

Hago la lectura completa de los datos.

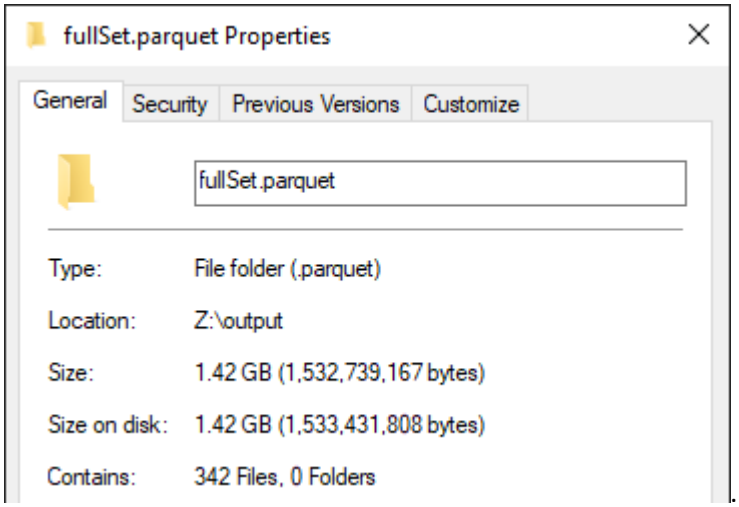
```
In [6]: df_fullSet = sqlContext.read.options(header='True', delimiter=',') \
            .schema(schema) \
            .csv("../bigdata/**")
```

3. Indexación de los datos para rápido acceso.

Una vez ingeridos los datos hacia un dataframe, se procedió a indexar y persistir estos datos hacia un almacenamiento columnar de apache parquet a fin de poder acelerar las consultas a su vez reducir aún mas el tamaño de los datos almacenados.

```
In [8]: df_fullSet.write.mode('ignore').parquet("./output/fullSet.parquet")
```

El RDD se almacenó en una colección de apache parquet cuyo tamaño se redujo de **10.4 GB** a **1.42 GB** agregando la posibilidad de indexación y rápida consulta sin pérdida de datos.



Como se explicó en secciones anteriores, el dataset contiene el estado de salud diario de cada uno de los discos duros que existen o existieron en el Datacenter. Esto significa que para una misma unidad de disco existe un registro diario de su estado de salud.
Es por esta razón que se realizó una query de relevancia la cual redujo la cantidad de tuplas del dataset a 1 tupla por unidad de disco duro que a su vez contenga la fecha en la cual esta unidad de disco comenzó a estar operativa **Min(Date)** y la fecha en la cual esta fue retirada **Max(Date)**.

Para ello la consulta agrupa por número de serie y almacena el primer día el cual ese disco estuvo operativo y el día máximo el cual el disco estuvo presente. Y calcula el tamaño del Disco en Terabytes.

```
Select
    Serial_Number,
    Model,
    Min(Date) as DayIn,
    Max(Date) as DayOut,
    CAST(FIRST(capacity_bytes/1099511627776) AS DECIMAL(10,2)) as SizeInTB,
    Count(1) as DayCount
from
    tbl_AllDiskData
group by Serial_Number,
    Model
order by Model
```

El nuevo esquema de datos es el siguiente.

```
|-- Serial_Number: string
|-- Model: string
|-- DayIn: date
|-- DayOut: date
|-- SizeInTB: decimal (10,2)
|-- DayCount: long
```

Con esta reducción. Procedimos nuevamente a almacenar este resultado en otra colección de Parquet en la cual basaremos la mayoría de nuestras futuras consultas.

Creo un dataframe con el total de discos Discos del Datacenter con su fecha de puesta en funcionamiento y dia de retiro.

```
In [12]: df_AllDisksDayInDayOut = sqlContext.sql('Select Serial_Number, Model, Min(Date) as DayIn, Max(Date) as DayOut, CAST(FIRST(capacit
```

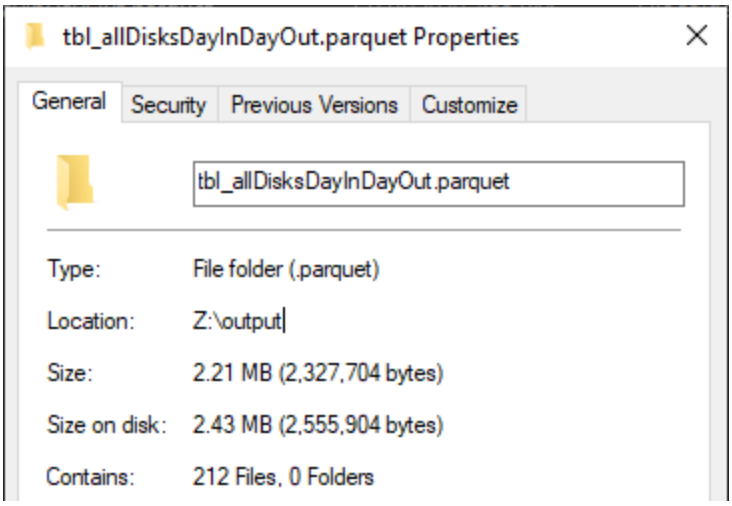
Guardo el resultado anterior como archivo de parquet comprimido para utilizar como de futuras consultas y asi obtener mayor velocidad para la realizacion de futuras consultas.

```
In [13]: df_AllDisksDayInDayOut.write.mode('ignore').parquet("./output/tbl_allDisksDayInDayOut.parquet")
```

Leo el resultado del parquet y lo cargo a un nuevo dataframe.

```
In [14]: df_allDisksDayInDayOut_pq = sqlContext.read.load("./output/tbl_allDisksDayInDayOut.parquet")
```

Esta reducción permitió que se pudiera trabajar con una colección de datos de 2.43 MBs



4. Creación de reportes y vistas con Pandas

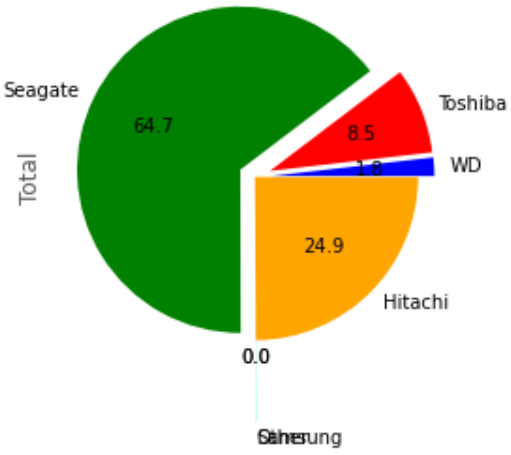
Para la creación de reportes de utilizó el módulo de PySpark SQL el cual permite hacer consultas utilizando un lenguaje conocido como lo es SQL y la librería de Pandas para poder graficar los resultados más importantes.

Se crearon los siguientes reportes:

Total de discos que existieron en el datacenter agrupados por modelo y fabricante

Model	Count
ST12000NM0007	38730
ST4000DM000	37035
ST12000NM0008	18470
TOSHIBA MG07ACA14TA	17476
HGST HMS5C4040BLE640	16346
ST8000NM0055	15006
HGST HUH721212ALN604	10917
ST8000DM002	10221
HGST HMS5C4040ALE640	8723
ST12000NM001G	6158
Hitachi HDS722020...	4774
ST3000DM001	4707
Hitachi HDS5C3030...	4664
Hitachi HDS5C4040...	2719
HGST HUH721212ALE600	2611
ST14000NM001G	2400
ST31500541AS	2188
ST6000DX000	1939
HGST HUH721212ALE604	1911
WDC WD30EFRX	1335

only showing top 20 rows



Maker	Total
Western Digital	4025
Toshiba	18844
Seagate	143144
Samsung	18
Other	100
Hitachi	55094

Era buena idea conocer con que unidades de discos y con que fabricante estaba mayormente integrado del Datacenter.

Para ello se creó una tabla auxiliar Makers que relaciona el fabricante con el modelo de disco duro. De tal forma que se pudiera hacer Join de los datos y obtener métricas de unidades de disco por fabricante.

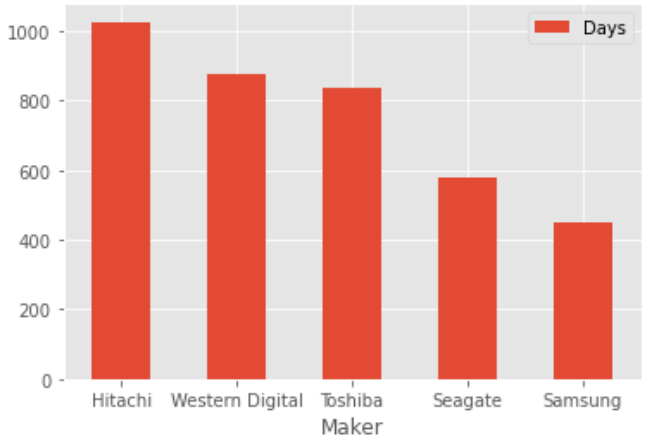
Se graficó con un gráfico de “torta” para que sea más visual el particionado de discos por fabricante.
Se puede observar gran cantidad de discos del centro de datos son del fabricante Seagate.

Cantidad promedio de días que un dura un disco duro en operación según su modelo, ordenados por cantidad de días en operación.

```
+----+--------------------+
|Days|               Model|
+----+--------------------+
|1888|      WDC WD5000BPKT|
|1753| TOSHIBA MD04ABA400V|
|1715|      WDC WD5000LPVX|
|1703|Hitachi HDS5C4040...|
|1637|         ST6000DX000|
|1591|      ST1000LM024 HN|
|1575|         ST4000DM000|
|1571|      WDC WD2500AAJS|
|1542|        WDC WD40EFRX|
|1530|       ST500LM012 HN|
|1507|HGST HMS5C4040ALE640|
|1491|         ST9320325AS|
|1487|Hitachi HDS5C3030...|
|1477|      WDC WD3200AAKS|
|1477|Hitachi HDS5C3030...|
|1446|      WDC WD5000LPCX|
|1433|HGST HMS5C4040BLE640|
|1428| TOSHIBA MD04ABA500V|
|1427|Hitachi HDS723030...|
|1423|Hitachi HDS723030...|
+----+--------------------+
only showing top 20 rows
```

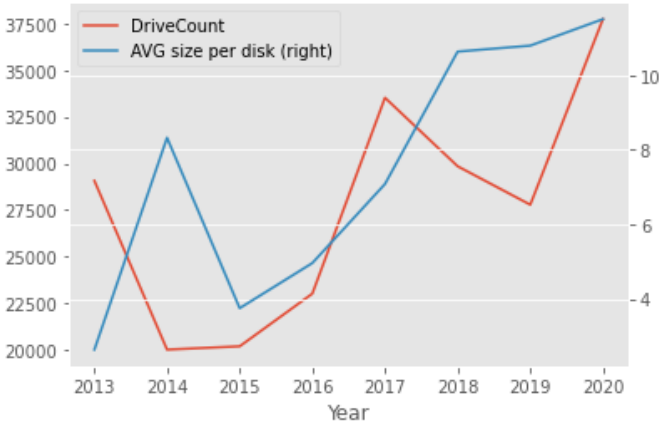
También era buena idea conocer cuál era la duración promedio de días que tenía un disco duro en el Datacenter y así poder determinar cuales eran los modelos más confiables hasta el momento Además conocer cuáles son los fabricantes más confiables.

```
+---------------+----+
|          Maker|Days|
+---------------+----+
|        Hitachi|1027|
|Western Digital| 876|
|        Toshiba| 835|
|        Seagate| 581|
|        Samsung| 449|
+---------------+----+
```



Cantidad de discos nuevos que ingresaron por año y tamaño aproximado de disco agregado.

También estudiar el crecimiento de discos duros que tuvo el Datacenter con el correr del tiempo y poder compararlo con el crecimiento en almacenamiento y así poder determinar como aumentó la densidad de almacenamiento por unidad de disco con el correr del tiempo.



Year	Count	SizeInTB	AVG size per disk
2013	29072	76776.20	2.64
2014	20035	166648.06	8.32
2015	20211	75692.28	3.75
2016	23030	114172.93	4.96
2017	33516	237143.00	7.08
2018	29849	317317.81	10.63
2019	27790	299931.30	10.79
2020	37723	433749.23	11.50

Cantidad de discos que se dieron de baja por año.

También era interesante saber cuántos discos fueron eliminados del Datacenter por año. Se dejó afuera el año 2020 ya que a la fecha no estaba completo el año

Year	Count
2013	1273
2014	5131
2015	4769
2016	6397
2017	13331
2018	15965
2019	11363

Cantidad de PB en el Datacenter por modelo de disco duro

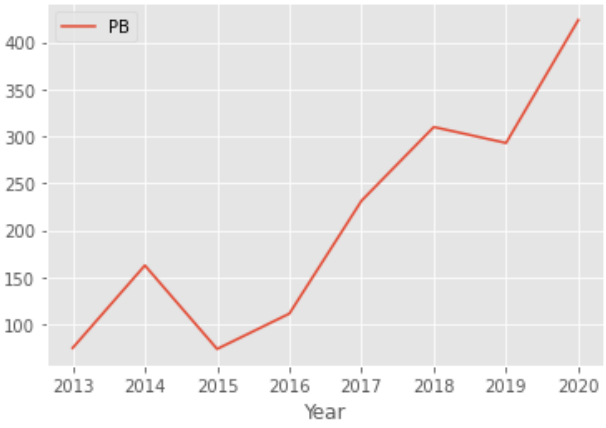
Estaba bueno conocer como la capacidad de almacenamiento se distribuía entre modelos de Discos

Model	PB	DriveCount
ST12000NM0007	412.63	38730
TOSHIBA MG07ACA14TA	217.25	17476
ST12000NM0008	196.78	18470
ST4000DM000	131.55	37035
HGST HMS5C4040ALE640	126.33	8723
HGST HUH721212ALN604	116.31	10917
ST8000NM0055	106.68	15006
ST8000DM002	72.67	10221
ST12000NM001G	65.61	6158
HGST HMS5C4040BLE640	58.11	16346
ST14000NM001G	29.84	2400
HGST HUH721212ALE600	27.82	2611
HGST HUH721212ALE604	20.36	1911
ST3000DM001	12.55	4707
Hitachi HDS5C3030...	12.43	4664
ST10000NM0086	11.09	1248
ST6000DX000	10.34	1939
Hitachi HDS5C4040...	9.67	2719
Hitachi HDS722020...	8.48	4774
HGST HUH728080ALE600	7.95	1118

only showing top 20 rows

Cantidad de PB agregados al Datacenter por año

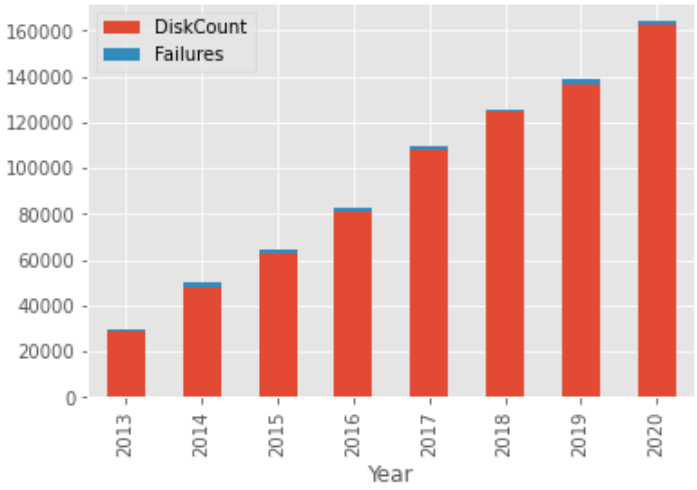
Estaba bueno conocer como crecía en capacidad de almacenamiento el Datacenter



Year	PB	DriveCount
2013	74.98	29072
2014	162.74	20035
2015	73.92	20211
2016	111.50	23030
2017	231.59	33517
2018	309.88	29849
2019	292.90	27790
2020	423.58	37723

Cantidad de discos que fallaron vs discos en el data center discriminado por año

Otro insight importante era ver cuál fue el porcentaje de fallas de disco duros en el transcurso de los años y así ver como aumentó la confiabilidad de los discos en el tiempo.



Year	DiskCount	Failures	Ratio
2013	29072	740	2.55
2014	47793	2206	4.62
2015	62898	1429	2.27
2016	81173	1431	1.76
2017	108282	1556	1.44
2018	124326	1381	1.11
2019	136568	2263	1.66
2020	162997	1093	0.67

5. Proyecciones con machine learning

Si bien, machine learning no fue parte de la materia, igualmente se realizaron proyecciones básicas utilizando machine learning a partir de las librerías linear_model de sklearn para generar predicciones lineales de algunos parámetros que consideré interesantes.

Cantidad de discos estimadas para el año 2021 y 2022

```
anio = regr.predict([[2021]])
print('Discos estimados para 2021:', int(anio))
```

```
Discos estimados para 2021: 179464
```

```
anio = regr.predict([[2022]])
print('Discos estimados para 2022:', int(anio))
```

```
Discos estimados para 2022: 198425
```

Fallas de discos calculadas para 2021

```
y_train = failuresVSDisks_pd['Failures'].values
x_train = np.array(failuresVSDisks_pd['Year'].values).reshape((-1, 1))
regr = linear_model.LinearRegression()
regr.fit(x_train, y_train)
y_pred = regr.predict(x_train)
anio = regr.predict([[2021]])
print('Discos a reemplazarse en 2021:', int(anio))
```

```
Discos a reemplazarse en 2021: 1659
```

Conclusiones.

Para cerrar el documento dividiré mis conclusiones en dos secciones:

Sobre los datos

En cuanto al análisis de los datos procesados, podemos decir que se pudo apreciar cómo con el avance de los años ha aumentado considerablemente la demanda por almacenar información. A su vez, Backblaze no es una empresa líder en esta industria, solo se limita a backups personales, con lo que es casi incalculable como ha aumentado la demanda en unidades de almacenamiento. Por otro lado, también podemos destacar que las unidades utilizadas en estos centro de datos provienen de los proveedores líderes de la industria y la calidad y confiabilidad de estos también ha mejorado con el avance de la tecnología. Esto se puede apreciar en la reducción en las fallas de estas unidades.

También, al evaluar cómo ha aumentado la densidad (capacidad promedio por unidad de disco) se puede deducir que hoy se requiere menos espacio, energía y refrigeración por terabyte, que hace 2 o 3 años atrás. Por último, haciendo una evaluación global de todos los años, se pudo determinar que el inventario de las unidades tiene una vida útil promedio de aproximadamente 2 años, siendo el fabricante Hitachi quien mayor vida útil posee, con un promedio de casi 3 años.

Sobre las herramientas

En cuanto al uso de herramientas para Big data. El poder trabajar con un enorme volumen de datos obteniendo latencias bajas fue crucial para poder analizar todos los datos de esta entrega.

A su vez, se apreció la potencia que brinda PySpark para el manejo sencillo de estos datos con lenguajes conocidos como lo es SQL. Y, si bien las Jupyter notebooks no son una herramienta exclusiva para trabajar con big data, la capacidad de tener un ambiente de desarrollo web que integre la posibilidad de almacenar resultados y mostrar gráficos fue el broche de oro para esta entrega.

En cuanto al uso de machine learning para el cálculo de predicciones, me quedó "gusto a poco" ya que si bien no fue parte de el dictado de esta materia y tenía nulos conocimientos, sin duda sembró una semilla para poder interiorizarme más en el tema.

Bibliografía.

- Blog de Backblaze.
  - https://www.backblaze.com/blog/category/cloud-storage/hard-drive-stats/
- “Propuesta de una Arquitectura de Gestión de Grandes Volúmenes de Datos para la Analítica en tiempo Real bajo Software Libre”, junio 2019. [Online]. Pedro Bonillo Ramos.
  - https://www.linkedin.com/pulse/propuesta-de-una-arquitectura-gesti%C3%B3n-grandes-datos-la-bonillo-ramos/
- Pandas API Reference.
  - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html#pandas.DataFrame.plot
- Matplotlib reference.
  - https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.pie
- Python base example for parsing files.
  - https://thispointer.com/python-how-to-get-list-of-files-in-directory-and-sub-directories/
- Repositorio del Proyecto.
  - https://github.com/matias53/BigData