Calcular em tempo real, a média de temperatura dos sensores IOT das máquinas da fábrica, a fim de monitoramento em tempo real da temperatura média de cada sensor para prevenção de danos às máquinas.

In [1]:
```
# Findspark
import findspark
findspark.init()
```

In [2]:
```
# Imports
import pyspark
from pyspark.streaming import StreamingContext
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, DoubleType
from pyspark.sql.functions import col, from_json
from time import sleep
```

In [3]:
```
# Conector para o Kafka
import os
os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0 pyspark-
```

In [4]:
```python
# Cria a sessão Spark
spark = SparkSession.builder.appName("AnaliseDeDadosDeSensoresIOT").getOrCreate()
```

```
:: loading settings :: url = jar:file:/Users/emerson/anaconda3/lib/python3.11/site-packages/pyspark/jar
s/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
```

```
Ivy Default Cache set to: /Users/emerson/.ivy2/cache
The jars for the packages stored in: /Users/emerson/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-b82cd3a1-00e3-46e8-9a6d-a57c5257c654;
1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.3.0 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.3.0 in central
        found org.apache.kafka#kafka-clients;2.8.1 in central
        found org.lz4#lz4-java;1.8.0 in central
        found org.xerial.snappy#snappy-java;1.1.8.4 in central
        found org.slf4j#slf4j-api;1.7.32 in central
        found org.apache.hadoop#hadoop-client-runtime;3.3.2 in central
        found org.spark-project.spark#unused;1.0.0 in central
        found org.apache.hadoop#hadoop-client-api;3.3.2 in central
        found commons-logging#commons-logging;1.1.3 in central
        found com.google.code.findbugs#jsr305;3.0.0 in central
        found org.apache.commons#commons-pool2;2.11.1 in central
:: resolution report :: resolve 669ms :: artifacts dl 24ms
        :: modules in use:
        com.google.code.findbugs#jsr305;3.0.0 from central in [default]
        commons-logging#commons-logging;1.1.3 from central in [default]
        org.apache.commons#commons-pool2;2.11.1 from central in [default]
        org.apache.hadoop#hadoop-client-api;3.3.2 from central in [default]
        org.apache.hadoop#hadoop-client-runtime;3.3.2 from central in [default]
        org.apache.kafka#kafka-clients;2.8.1 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.12;3.3.0 from central in [default]
        org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.3.0 from central in [default]
        org.lz4#lz4-java;1.8.0 from central in [default]
        org.slf4j#slf4j-api;1.7.32 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.8.4 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   12  |   0   |   0   |   0   ||   12  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-b82cd3a1-00e3-46e8-9a6d-a57c5257c654
        confs: [default]
        0 artifacts copied, 12 already retrieved (0kB/18ms)
```

```
24/02/15 13:39:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using
builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

Leitura do Kafka Spark Structured Stream

```
    Vamos começar conectando ao tópico criado no Kafka, abrindo uma inscrição para receber todos os
dados que o tópico receber.
    Criaremos StructTypes que serão as estruturas de dados que vão suportar o streaming e buscar a
temperatura, tendo como campos as chaves "leitura" e "temperatura".
    Após, será criado outro StructType para definição do esquema global, e será convertido cada linha
para String, para garantir o formato necessário para o parse (converter de JSON para estrutura da
análise). E com os dados garantidos no formato apropriado, será realizado o parse.
```

In [5]:
```python
# Cria uma subscrição no tópico do streaming de dados
df = spark \
  .readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "localhost:9092") \
  .option("subscribe", "IOT1") \
  .load()
```

In [6]:
```python
# Define o schema dos dados
esquema_dados_temp = StructType([StructField("leitura",
                                             StructType([StructField("temperatura", DoubleType(), True)])
```

In [7]:
```
# Define o schema global dos dados no streaming
esquema_dados = StructType([
    StructField("id_sensor", StringType(), True),
    StructField("id_equipamento", StringType(), True),
    StructField("sensor", StringType(), True),
    StructField("data_evento", StringType(), True),
    StructField("padrao", esquema_dados_temp, True)
])
```

In [8]:
```
# Captura cada linha dos dados como string
df_conversao = df.selectExpr("CAST(value AS STRING)")
```

In [9]:
```
# Parse do formato JSON em dataframe
df_conversao = df_conversao.withColumn("jsonData", from_json(col("value"), esquema_dados)).select("jsonDa
```

In [10]:
```
df_conversao.printSchema()
```

```
root
 |-- id_sensor: string (nullable = true)
 |-- id_equipamento: string (nullable = true)
 |-- sensor: string (nullable = true)
 |-- data_evento: string (nullable = true)
 |-- padrao: struct (nullable = true)
 |    |-- leitura: struct (nullable = true)
 |    |    |-- temperatura: double (nullable = true)
```

Preparando Data Frame

Vamos realizar um filtro pela coluna "padrao.leitura.temperatura" renomeando-a para "temperatura", e pela coluna "sensor", para facilitar o cálculo da média de temperatura por sensor.

```
In [11]: # Select das colunas
df_conversao_temp_sensor = df_conversao.select(col("padrao.leitura.temperatura").alias("temperatura"),
                                               col("sensor"))
```

```
In [12]: df_conversao_temp_sensor.printSchema()
```

```
root
 |-- temperatura: double (nullable = true)
 |-- sensor: string (nullable = true)
```

Análise de Dados em Tempo Real

Vamos criar todos os objetos antes da carga de dados, para que esteja tudo pronto no momento do acesso ao fluxo de dados em tempo real.
Será criado um objeto que agrupará os dados por sensor, e então será calculado a média de temperatura.
Como resultante, o nome da coluna recebe o prefixo de "avg", renomeamos a coluna para "media_temp" facilitando a análise.

```
In [13]: # Cálculo da média das temperaturas por sensor
df_media_temp_sensor = df_conversao_temp_sensor.groupby("sensor").mean("temperatura")
```

```
In [14]: df_media_temp_sensor.printSchema()
```

```
root
 |-- sensor: string (nullable = true)
 |-- avg(temperatura): double (nullable = true)
```

In [15]:
```
# Renomea as colunas
df_media_temp_sensor = df_media_temp_sensor.select(col("sensor").alias("sensor"),
                                                   col("avg(temperatura)").alias("media_temp"))
```

In [16]:
```
df_media_temp_sensor.printSchema()
```

```
root
 |-- sensor: string (nullable = true)
 |-- media_temp: double (nullable = true)
```

Análise em Tempo Real

```
    Será criado uma query de conexão ao streaming, criando uma tabela temporária ("IOT") completa e
colocando na memória para uma análise mais rápida.
    Para teste, será criado uma sequencia de 50 análises, de 3 em 3 segundos, retornando a média da
temperatura dos sensores, cujo a temperatura seja igual ou maior que 65 graus, buscando o
monitoramento das máquinas com temperaturas entrando na zona de atenção.
```

In [17]:
```
# Inicia a consulta ao streaming, criando tabela temporária, em memória
query_memoria = df_media_temp_sensor \
    .writeStream \
    .queryName("IOT") \
    .outputMode("complete") \
    .format("memory") \
    .start()
```

```
24/02/15 13:39:26 WARN ResolveWriteToStream: Temporary checkpoint location created which is deleted norm
ally when the query didn't fail: /private/var/folders/q_/r91lpwj123g_6kdpl4x8jv7r0000gn/T/temporary-9762
beaa-096f-423f-a60e-8f1bdadf7c2c. If it's required to delete it under any circumstances, please set spar
k.sql.streaming.forceDeleteTempCheckpointLocation to true. Important to know deleting temp checkpoint fo
lder is best effort.
24/02/15 13:39:26 WARN ResolveWriteToStream: spark.sql.adaptive.enabled is not supported in streaming Da
taFrames/Datasets and will be disabled.
24/02/15 13:39:28 WARN AdminClientConfig: The configuration 'key.deserializer' was supplied but isn't a
known config.
24/02/15 13:39:28 WARN AdminClientConfig: The configuration 'value.deserializer' was supplied but isn't
a known config.
24/02/15 13:39:28 WARN AdminClientConfig: The configuration 'enable.auto.commit' was supplied but isn't
a known config.
24/02/15 13:39:28 WARN AdminClientConfig: The configuration 'max.poll.records' was supplied but isn't a
known config.
24/02/15 13:39:28 WARN AdminClientConfig: The configuration 'auto.offset.reset' was supplied but isn't a
known config.
```

In [18]:
```
# Verifica streams ativados
spark.streams.active
```

Out[18]:
```
[<pyspark.sql.streaming.query.StreamingQuery at 0x10c00ef90>]
```

In [19]:
```python
# Query de execução em tempo real
for x in range(50):

    spark.sql("select sensor, round(media_temp, 2) as media from IOT where media_temp > 65").show()
    sleep(3)

query_memoria.stop()
```

```
+--------+-----+
|  sensor|media|
+--------+-----+
| sensor7|80.51|
|sensor34|84.55|
|sensor30|71.85|
| sensor4|73.68|
| sensor5|71.67|
|sensor28|71.69|
|sensor11|73.64|
|sensor35|79.15|
|sensor13|76.61|
|sensor32|69.28|
+--------+-----+

+--------+-----+
|  sensor|media|
+--------+-----+
| sensor7|80.51|
```