UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO DISTRIBUÍDOS

DANIEL FEITOSA DOS SANTOS - 11270591
IZABEL CHRISTINE DOS SANTOS BARRANCO - 11847711
GABRIELA COLOMBO ALVES LONGO - 11796812
LUAN CORRÊA MONTEIRO - 11797021
SILAS BOVOLIN REIS - 11796739

# **RELATÓRIO:** ANÁLISE DE DADOS COM APACHE SPARK

São Paulo
2022

# Sumário

# 1. Execução

Levando em consideração que a máquina para teste do EP será uma máquina linux GNU/Linux e com a distribuição Ubuntu 20.04.4 LTS, os passos de instalação e execução são específicos para máquinas Linux. Para a instalação em máquinas Windows existem algumas especificidades.

## 1.1 Pré-Requisitos

Como pré-requisitos para rodar o projeto é necessário possuir o Python e o Java instalados. Idealmente a versão do Java baixada deve ser a 11 e a versão do python atualizada. Caso necessário, as variáveis de ambiente do java e python devem ser configuradas de acordo e predefinidas na variável path, para que assim os comandos do java e python sejam reconhecidos diretamente em qualquer pasta no terminal do Linux.

## 1.2 Instalação

Utilizamos o gerenciador de pacotes do Python, o pip, para instalar todas as dependências que o projeto funcional requer para ser executado. Dessa forma, após descompactar o arquivo submetido e estar na pasta raiz do projeto, execute o seguinte comando: `pip install -r requirements.txt` para instalar os pacotes **antes** de executar de fato o programa. Caso você tenha o Python3 instalado na sua máquina, mas o que é utilizado por padrão seja o Python2.7 (comum de ocorrer em sistemas UNIX-based), você pode tentar os seguintes comandos alternativos:

```
pip3 install -r requirements.txt
```

```
python3 -m pip install -r requirements.txt
```

Caso mesmo assim você ainda não consiga rodar o pip, recomendamos que acesse essa página sobre ambiente virtual utilizando pip e tente novamente.

### 1.2.1 Spark

Será necessário baixar o Spark para assim conseguir utilizar a biblioteca pyspark. O spark pode ser baixado através da seguinte página. Utilizamos da versão 3.2.1 (Jan 26 2022) do spark, porém a versão mais recente deve funcionar também.

Após descompactar o Spark, será necessário configurar o caminho do spark na variável de ambiente path para que assim ele seja reconhecido pelo pyspark.

## 1.3 Como executar?

Foram criados arquivos .py para cada uma das questões perguntadas pelo enunciado do EP. Os arquivos seguem a seguinte nomenclatura e se referem às seguintes questões:

| Nome | Questão |
|---|---|
| q1.py | “Como é a requisição de recursos computacionais (memória e CPU) do cluster durante o tempo? ” |
| q2.py | “As diversas categorias de jobs possuem características diferentes (requisição de recursos computacionais, frequência de submissão, etc.)? ” |
| q3.py | “Quantos jobs são submetidos por hora? ” |
| q4.py | “Quantas tarefas são submetidas por hora?” |
| q5.py | “Quanto tempo demora para a primeira tarefa de um job começar a ser executada?” |
| q6.py | Qual a frequência de cada tipo de job para os eventos de job? |
| q7.py | Qual a frequência de cada tipo de job para os eventos de tarefas? |

As questões 6 e 7 não estavam presentes no enunciado do EP, porém foram adicionadas para a nossa análise dos dados disponibilizados.

Estando com todas as dependências e a instalação do python, java e spark feita, a execução pode ser feita a partir do seguinte comando no terminal: “python *caminho-do-arquivo*”, sendo *caminho-do-arquivo* o caminho do arquivo .py a ser executado. Antes da execução desses arquivos é importante que o caminho para os dados seja configurado corretamente, nas linhas iniciais de cada um dos arquivos o caminho dos dados é configurado, portanto este caminho deve ser configurado de acordo com o caminho dos dados em sua máquina. Ao executar um desses arquivos serão imprimidos na tela os resultados obtidos para a questão a qual o arquivo se refere, sendo gráficos ou objetivamente valores referentes ao resultado.

Caso seja preferível visualizar os resultados diretamente, eles se encontram na pasta *\results* dentro do arquivo .zip do projeto.

# 2. Análises

Com base nos dados dos traços fornecidos foi possível realizar algumas análises com a ajuda do Spark com a finalidade de conhecer um pouco mais sobre o comportamento do ambiente de execução e das tarefas executadas no aglomerado do Google.

Para responder às questões propostas para este trabalho, a todo momento iremos fazer referências a tipos específicos de tarefas de acordo com a sua numeração, que já é documentada e mapeada pelo próprio Google. Segue a tabela com a numeração e o significado de cada uma das tarefas.

| Tipo | Nome do tipo | Significado segundo a documentação |
|---|---|---|
| 0 | SUBMIT | “The collection or instance was submitted to the scheduler for scheduling.” |
| 1 | QUEUE | “The collection or instance was marked not eligible for scheduling by the batch scheduler.” |
| 2 | ENABLE | “The collection or instance became eligible for scheduling.” |
| 3 | SCHEDULE | “The collection or instance started running.” |
| 4 | EVICT | “The collection or instance was descheduled because of a higher priority collection or instance, or because the scheduler overcommitted resources.” |
| 5 | FAIL | “The collection or instance was descheduled due to a failure.” |
| 6 | FINISH | “The collection or instance completed normally.” |
| 7 | KILL | “The collection or instance was cancelled by the user or because a depended-upon collection died.” |
| 8 | LOST | “The collection or instance was presumably terminated, but due to missing data there is insufficient information to identify when or how.” |
| 9 | UPDATE_PENDING | “The collection or instance was updated (scheduling class or resource requirements) while it was waiting to be scheduled.” |
| 10 | UPDATE_RUNNING | “The collection or instance was updated while it was scheduled somewhere.” |

Além disso, também para efeito expositivo, abaixo segue uma imagem que explica de maneira intuitiva e visual o ciclo de vida dos *jobs* e das tarefas que são monitoradas pelo gerenciador de cluster do Google.

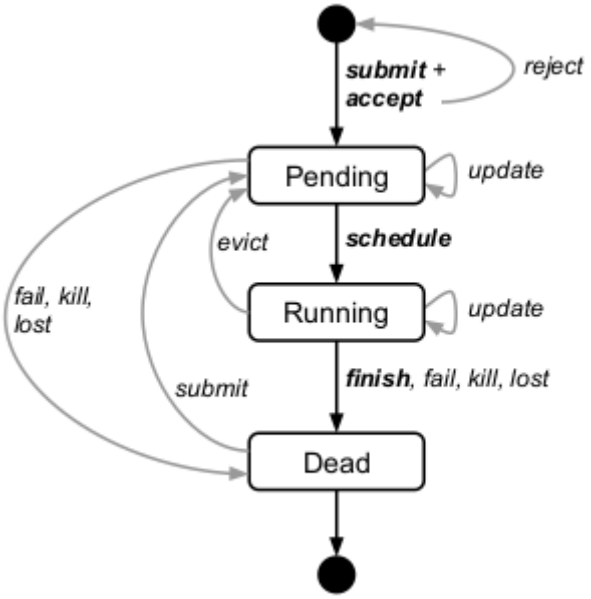



**Figura 1:** Diagrama de estados de Jobs e de Tasks. Usuários (desenvolvedores e sysadmins do Google) podem "engatilhar" (fazer *trigger*) de transações de *submit*, *kill* e *update*.

## 2.1 Requisição de recursos computacionais durante o tempo

*"Como é a requisição de recursos computacionais (memória e CPU) do cluster durante o tempo? "*

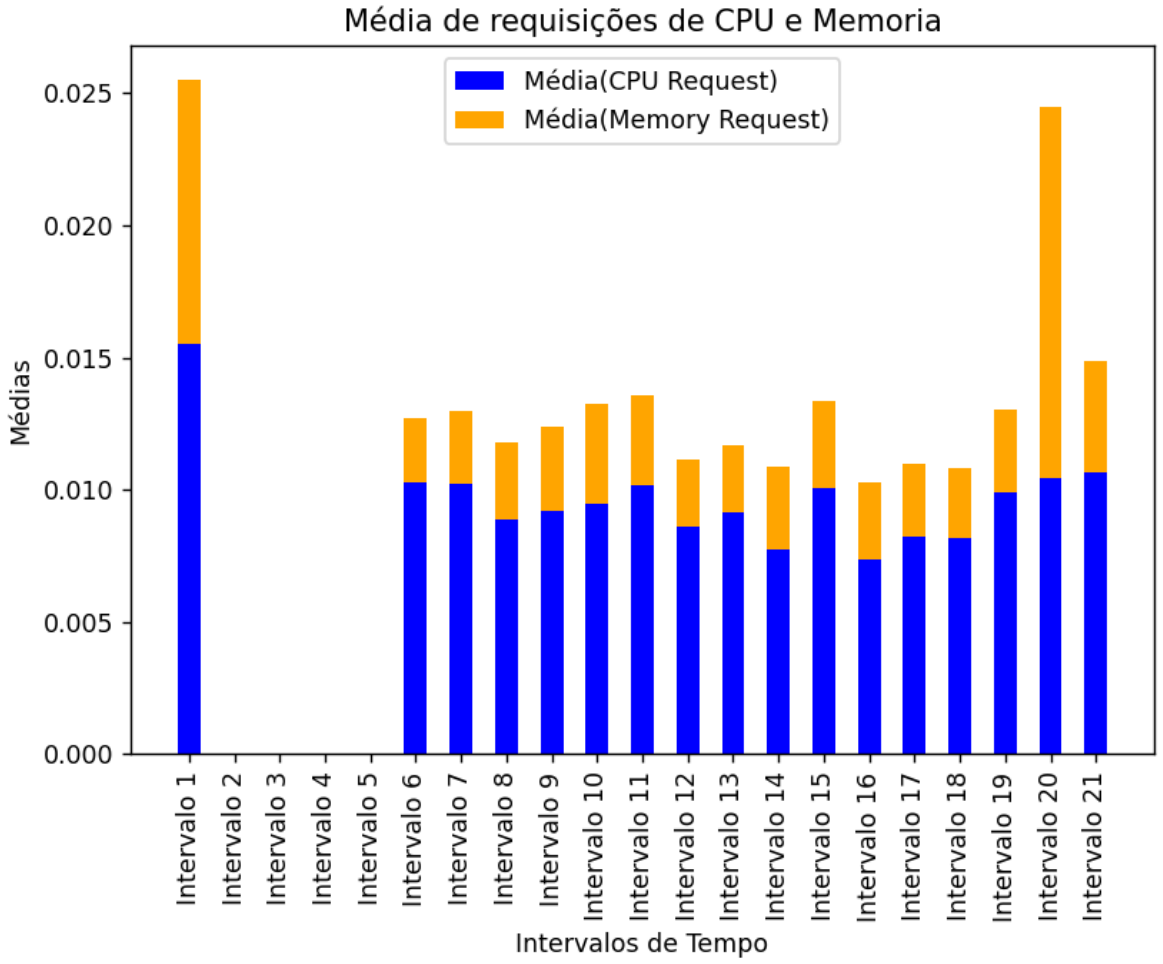
Para que conseguíssemos analisar a requisição de recursos durante o tempo obtivemos gráficos que representam as médias de requisições de CPU e memória durante intervalos de tempo de mesmo tamanho.

Considerando que os traços de execução capturados são referentes a uma semana, separamos os dados em 21 intervalos de tempo, para assim teoricamente contemplar 3 períodos diários. Dessa forma seria mais aparente se houvesse diferenças periódicas entre a requisição de CPU e memória durante os períodos do dia.

No gráfico abaixo é possível observar que do intervalo de tempo 2 ao 5 a média da coluna referente a requisições de recursos de CPU e memória se aproxima de 0. Durante o intervalo de tempo 1 e 20 houve um pico de uso de memória não usual quando comparado com os outros intervalos, durante o intervalo 1 também houve um pico de uso de requisição de usos de CPU não usual. Uma possível explicação para o alto consumo de memória durante

o intervalo de tempo 1 é o fato da execução ter acabado de ser iniciada, assim sendo necessário carregar todos os dados possíveis que serão usados durante a execução.

**Gráfico 1 - Média de CPU e memória dos eventos das tarefas de acordo com os intervalos de tempo**




A requisição de recursos computacionais(CPU e memória) ocorre de maneira uniforme(com pequenas variações) para a maior parte do tempo no cluster, com exceção em alguns intervalos de tempo em que o cluster aparenta não estar usando dos recursos computacionais disponíveis e em outros intervalos de tempo como o intervalo 1 e 20 em que a requisição para o uso de memória foi elevado.

**Gráfico 2 - Média de requisição de CPU  dos eventos de tarefas de acordo com os intervalos de tempo**

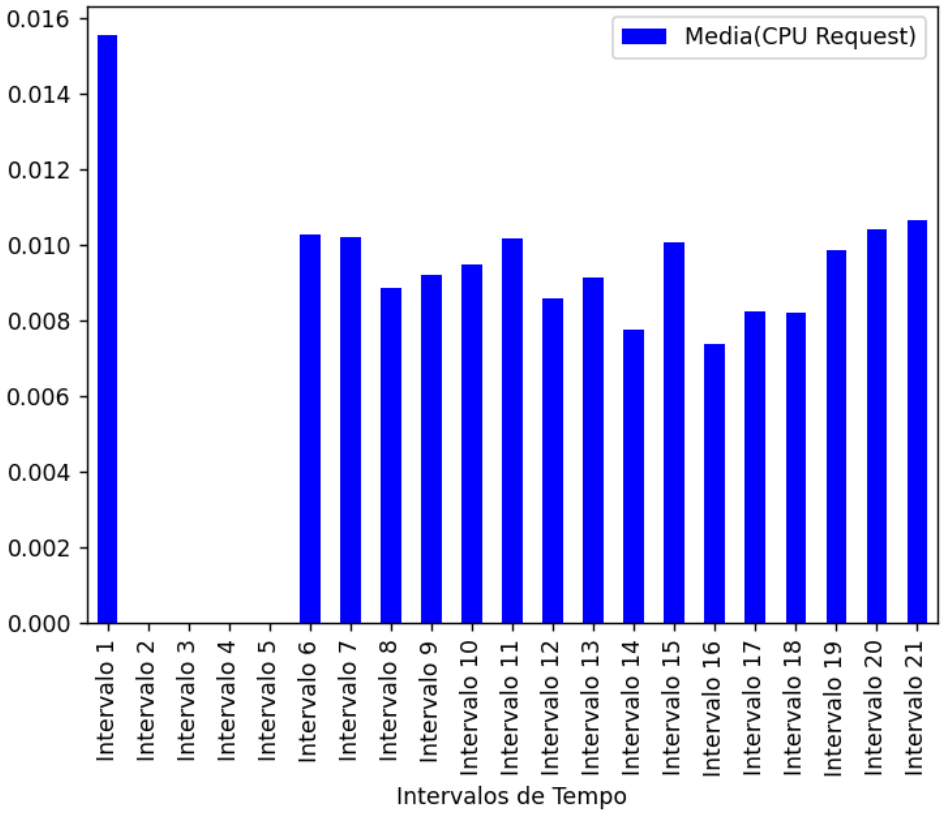
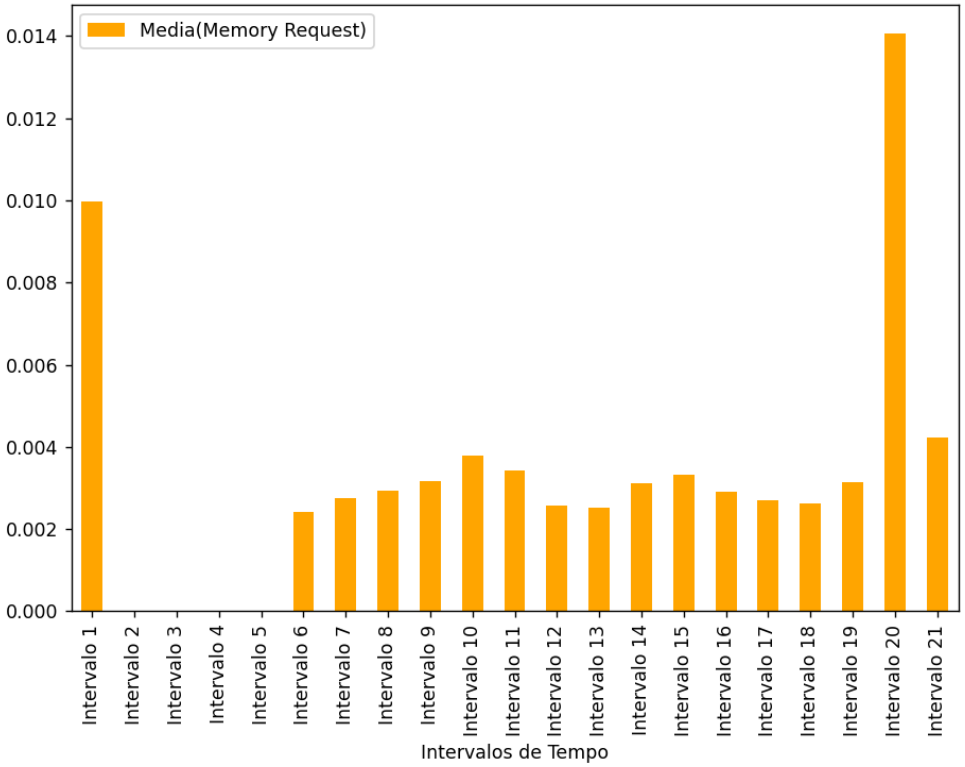



**Gráfico 3 - Média de requisição de memória  dos eventos de tarefas de acordo com os intervalos de tempo**



## 2.2 Características das categorias de jobs

*"As diversas categorias de jobs possuem características diferentes (requisição de recursos computacionais, frequência de submissão, etc.)?"*

Para realizar a categorização de *jobs* foi utilizado o atributo *priority*, com a seguinte relação:

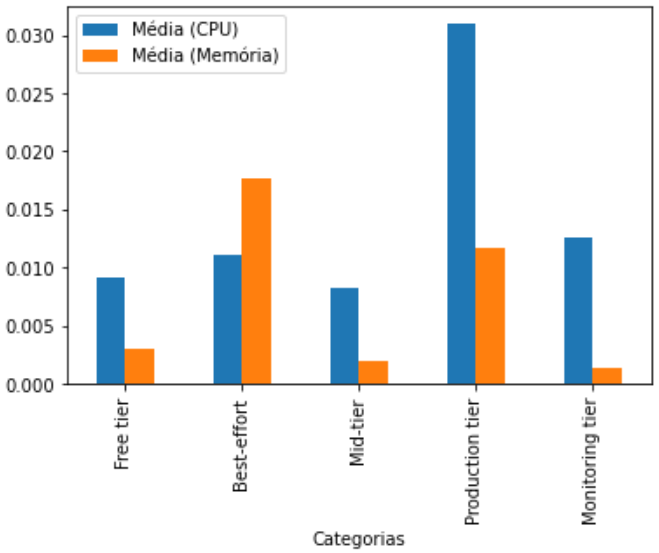
- Free tier: priority ≤ 99
- Best-effort Batch: 100 ≤ priority ≤ 115
- Mid-tier: 116 ≤ priority ≤ 119
- Production tier: 120 ≤ priority ≤ 359
- Monitoring tier: priority ≥ 360

Na primeira parte da questão foi verificado o uso do processamento da CPU e da memória principal de acordo com cada categoria. Com isso, foi construído um gráfico que mostre a diferença entre as categorias, comparando a média de requisição de recursos da CPU e da memória principal. Nesse gráfico é visto como conclusão que a categoria *Production Tier,* que tem a prioridade entre 120 e 359, a média de uso da CPU é mais significativa que em relação às demais, porém a categoria *Mid Tier* tem a menor média comparado com as outras categorias.

Ainda assim, quando é comparado ao uso de Memória a prioridade entre 100 e 115, representando a categoria *Best Effort*, tem um aumento considerável se comparado às outras categorias. Já a categoria *Monitoring Tier*, representando o nível de prioridade maior que 360, apresenta o nível mais baixo de memória.

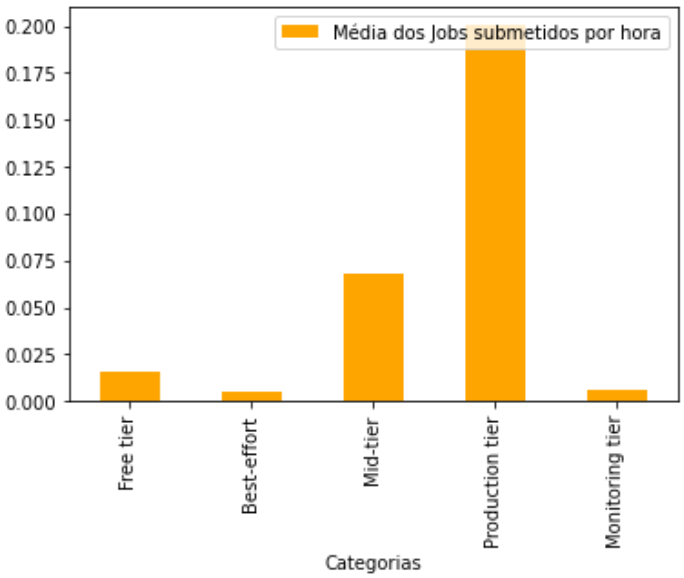
Abaixo é possível observar como cada uma das categorias se comporta em relação ao uso de CPU e Memória.

**Gráfico 4 - Comparativo entre categorias de acordo com a CPU e a Memória**




Ademais, foi criado uma função para ver a quantidade de *jobs* submetidos por cada categoria. Para isso, foi construído um gráfico com todos os resultados de cada categoria, e concluímos que a categoria *Production Tier* com prioridade entre 120 e 359 apresenta uma média de *jobs* submetidos maior do que as demais. Portanto, pode-se concluir que essa categoria submete um número significativo de *jobs* por hora quando comparado com as demais categorias.

**Gráfico 5 - Comparativo entre categorias de acordo com os jobs submetidos**




Considerando a pergunta realizada no enunciado, as diversas categorias de jobs possuem características diferentes como observado nos gráficos acima. Enquanto o production tier é um dos mais demandantes em relação a recursos computacionais, o free tier e o mid-tier utilizam menos recursos computacionais.

## 2.3 Submissão de jobs

"Quantos jobs são submetidos por hora? "

A finalidade desta análise era ter conhecimento de quantos *jobs* são submetidos por hora, portanto foi calculada a média de *jobs* submetidos por hora. Como resultado, obtivemos que a média de *jobs* submetidos por hora foi de 0,2935 com um total de *jobs* submetidos de 175.098.

## 2.4 Submissão de tarefas

*"Quantas tarefas são submetidas por hora?"*

Para responder a esta pergunta calculamos a média de tarefas submetidas por hora. Como resultado, a média de tarefas submetidas por hora foi de 127,949 com um total de *tarefas* submetidas de 76.323.734.

## 2.5 Primeira tarefa de um job

*"Quanto tempo demora para a primeira tarefa de um job começar a ser executada?"*

A análise realizada se deu com base na quantidade de tempo que leva para a primeira tarefa de um *job* começar a ser executada. A média de tempo para as primeiras tarefas começarem a ser executadas obtida foi 2,967 segundos e o desvio padrão foi de 1,909. Portanto demora em média aproximadamente 3 segundos para as primeiras tarefas de um *job* começarem a ser executadas. A partir do desvio padrão é possível concluir que os tempos para as primeiras tarefas de um *job* começarem a ser executadas variam ao redor da média, porém não variam de maneira exagerada, assim estando em sua maioria em torno de 3 segundos.

## 2.6 Quantidade de tarefas por tipo

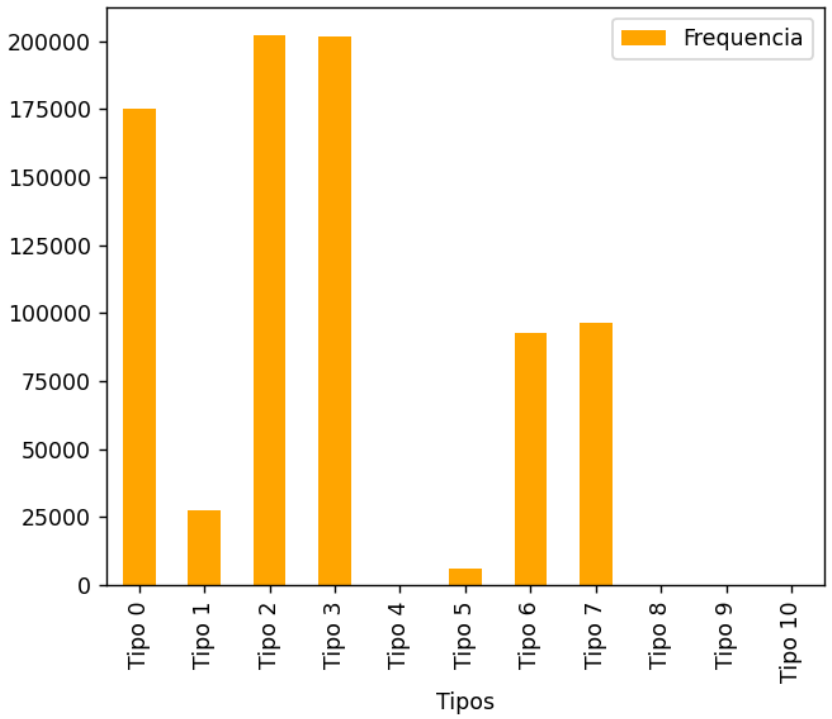
*Qual a frequência de cada tipo de job para os eventos de job?*

Os *Jobs* são conjuntos de tarefas/recursos que podem ser alocadas. Com isso, os jobs podem estar em diferentes estados, que são citados em análises. Nos traços de execução do Google Borg esses estados são descritos pelos diferentes tipos de eventos de jobs. Pensando

nisso, foi construído um gráfico que mostra a frequência desses eventos para os diferentes tipos de eventos.

Abaixo é mostrado o gráfico de frequência de cada evento, dessa forma pode-se concluir que existem mais eventos dos *jobs* com os seguintes tipos: o tipo 0 que representa a submissão do *job*, o tipo 2 que vai focar na programação de quando o *job* vai ser executado, e o tipo 3 que ocorre quando o *job* é "bloqueado" para outra instância com maior prioridade poder executar.

**Gráfico 6 - Frequência de cada eventos dos jobs(histograma)**

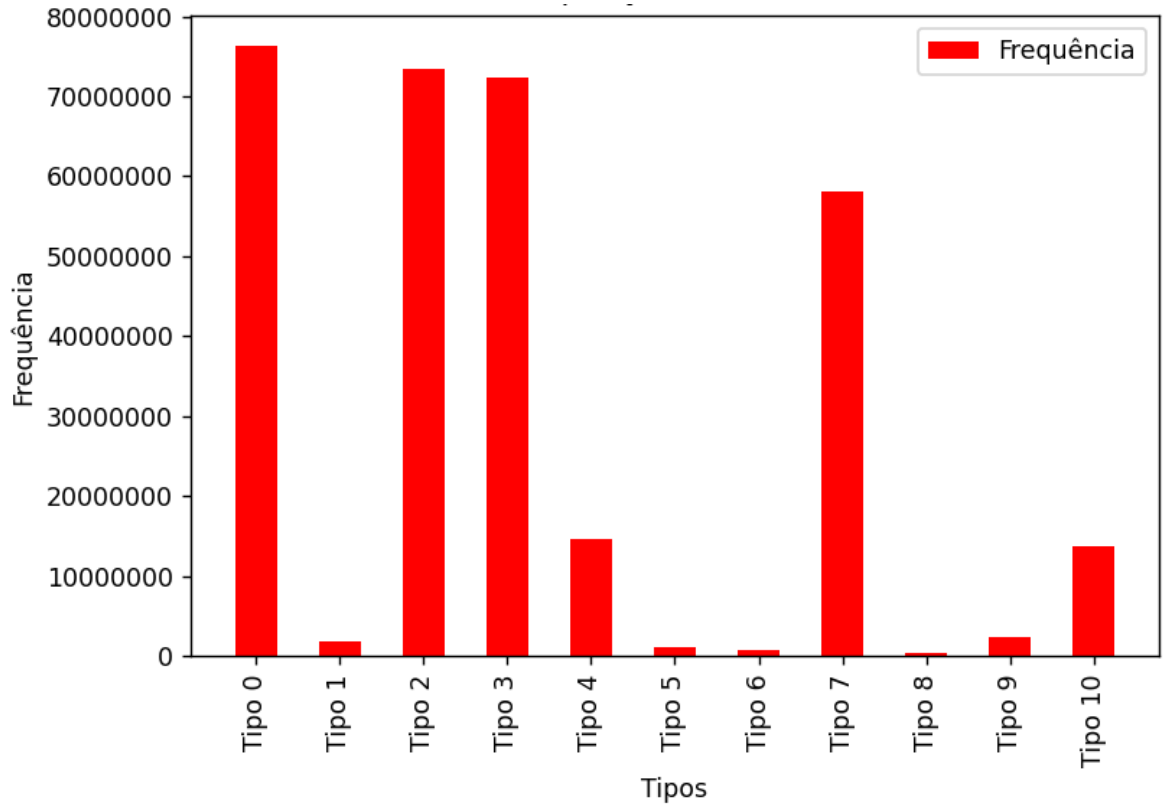



## 2.7 Quantidade de tarefas por tipo

*Qual a frequência de cada tipo de job para os eventos de tarefas?*

Para cada um dos tipos de eventos foi obtida a frequência dos eventos das tarefas, para assim produzirmos um histograma. Com o histograma abaixo pode-se concluir que os 3 maiores tipos de eventos de tarefas são: tipo 0 com a submissão da tarefa junto ao *job*, o tipo 2 referenciando a programação para começar a executar aquela determinada tarefa, e o tipo 3 que vai buscar priorizar aquela outra tarefa que tem mais prioridade, bloqueando aquela que estava sendo executada anteriormente.

**Gráfico 7 - Frequência de cada eventos das tarefas(histograma)**




# 3. Implementação

A implementação se deu de forma separada para cada análise a ser feita, ou seja, cada análise possui um arquivo a ser executado separadamente. A nomenclatura desses arquivos pode ser verificada em: execução.

## 3.1 Requisição de recursos computacionais

*"Como é a requisição de recursos computacionais (memória e CPU) do cluster durante o tempo? "*

A implementação das consultas para responder essa questão se deu utilizando a tabela *"instance_events"* com a finalidade de analisar a requisição de recursos computacionais do cluster durante o tempo.

Os dados foram filtrados por eventos do tipo 3, que se referem a tarefas de jobs do tipo *schedule*. Posteriormente separamos os eventos em intervalos de tempo obtidos através de uma filtragem e para cada um desses intervalos de tempo calculamos a média de execução.

Para podermos calcular intervalos de tempo que estivessem dentro do período de tempo em que os traços de execução se encontravam, foi verificado o tempo máximo e mínimo dos eventos, assim foi possível determinar o intervalo de tempo em que os traços de

execução se referiam e portanto foi possível calcular os intervalos de tempo que se encaixavam na janela de tempo a que os traços de execução se referiam. Para a nossa análise os eventos das tarefas foram divididos em 21 intervalos de tempo.

Para cada intervalo de tempo, é calculada a média dos recursos de memória e CPU que estão sendo alocados para cada uma das tarefas. Ao final, foi plotado um gráfico contendo as médias de requisição de CPU e memória de cada intervalo.

## 3.2 Características das categorias de jobs

*"As diversas categorias de jobs possuem características diferentes (requisição de recursos computacionais, frequência de submissão, etc.)?"*

Para responder a esta questão foram utilizados as seguintes tabelas *"collection_events"* e *"instance_events"*. Com isso, foi verificado as 5 categorias dos *jobs* e das tarefas para análises posteriores. Os *dataframes* são filtrados de acordo com suas prioridades.

Na primeira parte da consulta, utilizando os dados das tarefas foi construído uma função que vai descrever cada categoria, de acordo com os atributos *"resource_request_cpus"* e *"resource_request_memory"*, mostrando a variância, média, desvio padrão, máximo e mínimo de cada categoria.

Já na segunda parte da consulta, utilizando os dados dos *jobs*, foi desenvolvida uma função que filtra o tipo 0 de cada categoria, com o objetivo de analisar apenas os *jobs* já submetidos. A partir disso, é agrupado as cinco categorias com o máximo e o mínimo de tempo, buscando o primeiro e o último job que foi submetido, para depois buscar o intervalo de tempo para executar e job. E por último, é realizada a média desses *jobs* submetidos de acordo com cada categoria.

No resultado para essa segunda parte da consulta foi percebido que a categoria *"production_tier"* tem uma média maior de *jobs* submetidos por hora do que as demais, e a categoria *"best_effort"* tem uma média inferior às demais.

## 3.3 Submissão de jobs

*"Quantos jobs são submetidos por hora?"*

Para responder esta questão foram utilizados os arquivos *"collection_events"*. Como a média foi a medida utilizada para a análise, foi filtrado pelo tipo 0 que indica os *jobs* submetidos, com esse filtro foi contabilizada a quantidade de *jobs* submetidos e buscado o

máximo e o mínimo de tempo do primeiro e último job submetidos, para assim calcular o intervalo de tempo. Com essas informações, foi possível realizar o intervalo entre esses tempos e assim fazer a média da seguinte forma: $\frac{jobs\ submetidos}{intervalo\ de\ tempo\ /\ 3600}$.

## 3.4 Submissão de tarefas

*"Quantas tarefas são submetidas por hora?"*

Para esta questão foram utilizados os arquivos *"instance_events"*. Para responder a questão calculamos a média da seguinte forma: filtramos os eventos pelo tipo 0, para que assim obtivéssemos o total de tarefas submetidas através da função de agregação *count.* Além disso, nessa coleção foi buscado pela tarefa com o menor timestamp e a com o maior com o intuito de calcular o intervalo de tempo entre essas duas instâncias, para que assim fosse calculada a média, sendo a média de tarefas submetidas por hora: $\frac{tarefas\ submetidos}{intervalo\ de\ tempo\ /\ 3600}$.

## 3.5 Primeira tarefa de um job

*"Quanto tempo demora para a primeira tarefa de um job começar a ser executada?"*

Para responder a esta questão foram utilizados os arquivos *"instance_events"*. Para saber o tempo necessário para a primeira tarefa de um job começar a ser executada, a nossa consulta começa filtrando as tarefas que começam a serem executadas para em seguida agrupar pelo *"collection_id"* - onde é possível identificar tarefas de um mesmo job - e buscando o menor tempo de cada job, que seria a primeira tarefa a ser executada dentro do job. Com esses tempos mínimos, algumas medidas estatísticas foram obtidas.

## 3.6 Quantidade de jobs por tipo

*Qual a frequência de cada tipo de job para os eventos de job?*

Para esta questão, a consulta feita utiliza da tabela de eventos *"collection_events"*. É utilizado do método *groupBy* para agrupar os dados pelo tipo e assim utilizar da função de agrupamento *count* para obter a quantidade dos eventos por tipo. Em seguida os dados são colocados em um arranjo para que assim seja possível plotar um histograma demonstrando a frequência dos eventos por tipo.

## 3.7 Quantidade de tarefas por tipo

*Qual a frequência de cada tipo de job para os eventos de tarefas?*

A implementação das consultas que respondem a esta pergunta foi elaborada de forma similar a consulta feita para a pergunta anterior, porém ao invés da tabela *“collection_events” foi usada a tabela “instance_events”,* assim foi possível observar com qual frequência dos eventos de tarefas para cada tipo de job.

# 4. Conclusão

Com o auxílio do Apache Spark foi possível processar de forma distribuída os dados de traço de execução no aglomerado do Google. Como os arquivos de traços de execução possuem um tamanho grande, se caso fosse utilizado outra ferramenta para o processamento, isto decerto, tomaria um tempo de execução das consultas bem maior do que o necessário tomado com o uso do Apache Spark.

Por fim, ao processar os dados foi possível conhecer um pouco mais sobre o comportamento de um cluster do Google, durante um certo período de tempo, a respeito de dados das tarefas e *jobs* submetidos e executados, contabilizando seus tempos, requisições uso de recursos computacionais, como memória e CPU e frequências de submissões.

# 5. Referências bibliográficas

VERMA, Abhishek. et. al. **Large-Scale cluster management at Google with Borg**. In: European Conference on Computer Systems (EuroSys), Bordeaux, France, 2015. Google Inc.

WILKES, John. et. al. **Google cluster-usage traces v3**. 2020. Google Inc.