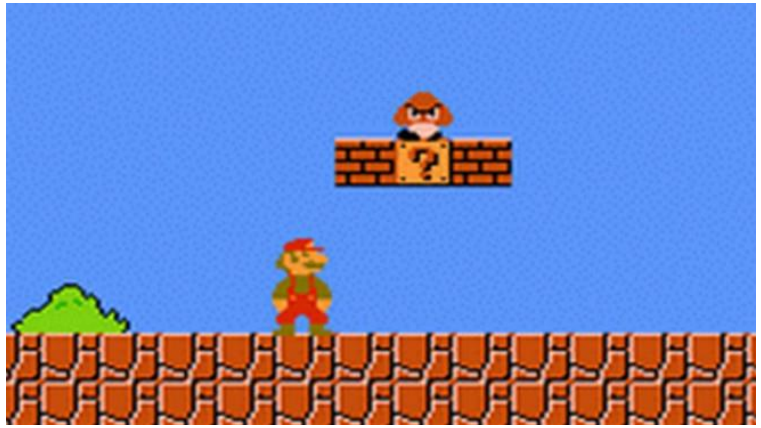
Cacheamento Lado Servidor

O processamento mais rápido é aquele que não é feito

Que Mário?

- Desenvolvedor há 15 anos
- Saindo da caixinha há 3 anos
- Foco em soluções para Contact Center
- Trabalha na G4 Solutions Software
- jmarioguedes@gmail.com



Operações Idempotentes

“Em matemática e ciência da computação, a idempotência é a propriedade que algumas operações têm de poderem ser aplicadas várias vezes sem que o valor do resultado se altere após a aplicação inicial.”

https://pt.wikipedia.org/wiki/Idempot%C3%AAncia

http://www.infoq.com/br/news/2013/05/idempotent

Pegando carona na palestra do Diego Garcia...

- Em sistemas RESTful operações GET deveriam ser *idempotentes.*
- Em uma arquitetura baseado em micro serviços o resultado produzido por uma instância pode ser reutilizado por outra.
- Ou seja, se a mesma chamada for feita uma ou mais vezes sob as mesmas condições, a resposta será a mesma.
- *Por que não guardar esta resposta para reutilização?*

O que poderia ser cacheado?

- Resultados de processamentos, em tipos Python mesmo, serializados com *pickle*
- Imagens, especialmente as processadas como em redimensionamentos
- Listagens de consultas ao banco de dados
- E por ai vai ...

Mas isso pode virar um inferno!

- Deve-se invalidar o cache, especialmente em operações CRUD
- Um descuido e ficaremos horas procurando um problema que não existe.
- Recomendado determinar um tempo de vida, renovando-se quando necessário

REDIS – REmote DIctionary Server

- Banco de Dados noSQL baseado em **chave\valor**
- Pode ser utilizado como serviço de mensageria (Filas)
- Possui a facilidade PUB\SUB
- http://redis.io/
- Cliente Python: https://pypi.python.org/pypi/redis

Não encontramos nada pronto... Existe?

- Mas beleza, o Luciano Ramalho me ensinou o que é decorator ☺
- https://adm.python.pro.br/
- Vamos a um exemplo simples

```python
from pickle import dumps, loads
from redis import Redis
from pymongo import Connection
from time import time

REDIS = Redis('192.168.1.60')
MONGO = Connection('192.168.1.60', 64102)

REDIS.flushall()


# MONGO['grupy']['exemplo'].remove({})
# for i in range(0, 1000000):
#     MONGO['grupy']['exemplo'].insert({'nome': 'usuario_{}'.format(i), 'equipe': 'ALPHA'})


def caching(metodo):
    def buscar_ou_processar(*args, **kwargs):
        chave = metodo.__name__ + ':' + ':'.join([str(item) for item in args] + [str(valor) for valor in kwargs.values()])
        buffer = REDIS.get(chave)
        if not buffer:
            buffer = metodo(*args, **kwargs)
            if buffer:
                REDIS.set(chave, dumps(buffer))
                return buffer
        else:
            return loads(buffer)

    return buscar_ou_processar


@caching
def usuarios_associados(nome_equipe: str):
    documentos = MONGO['grupy']['exemplo'].find({'equipe': nome_equipe})
    return {'usuarios': [doc['nome'] for doc in documentos], 'quantidade': documentos.count()}


if __name__ == '__main__':
    agora = time()
    usuarios_associados(nome_equipe='ALPHA')
    print('Tempo', time() - agora)

    agora = time()
    usuarios_associados(nome_equipe='ALPHA')
    print('Tempo', time() - agora)
```

O exemplo foi absurdo justamente para mostrar a importância do assunto

C:\Python34\python.exe "C:\Program Files (x86)\JetBrains\PyCharm 4.5\helpers\pydev\pydevd.py" --multiproc --client 127.0.0.1 --port 50827 --file F:/GruPy/exemplo.py

pydev debugger: process 5628 is connecting

Connected to pydev debugger (build 141.1899)

1ª vez | Tempo **123**.25300002098083

2ª vez | Tempo **19**.740999937057495

Process finished with exit code -1

Pontos de atenção!

- Só vale a pena se o método em questão depender de comunicação com outros softwares como banco de dados por exemplo.
- O Python oferece o decorator `functools.lru_cache`
- Cuide da invalidação de cache, pela sua sanidade mental.

Obrigado!

- *Em memória de Wanc Guttemberg, com quem aprendi muito pelos códigos da vida.*