Sistemas Paralelos e Distribuídos

Práticas – Aula 10

Práticas

Middleware em sistemas distribuídos

- Intro
- Conceitos
- Cache em memória
- Exemplos com Redis
- Broker System
- Exemplos com Rabbitmq
- Conclusões

Intro

- Termo utilizado pela primeira vez em conferência da NATO em 1968.

- Inicialmente projetado para a comunicação de sistemas monolíticos.

- Atualmente, o papel do middleware em sistemas distribuídos é de introduzir uma camada de abstração. Em geral entre o sistema operativo ou sistema principal e a interface do usuário.

- Promove a simplificação da gestão do sistema e suas integrações.

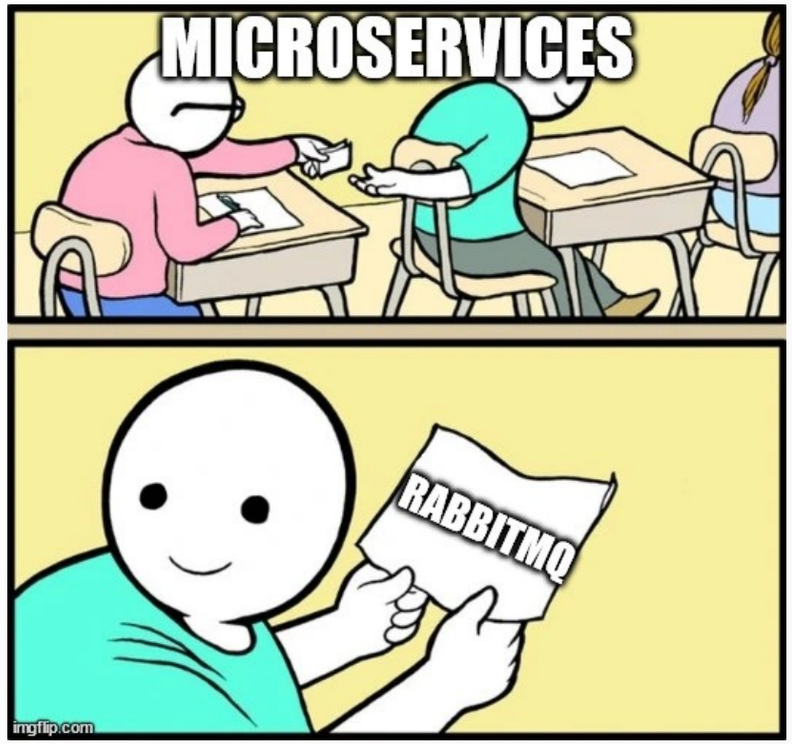
- De certa forma o middleware acaba por facitliar o desenvolvimento de software para SD.

- Exemplos de middleware podem incluir

    - Message queues
    - RPC
    - Event notifications
    - Brokers
    - Cache
    - Service Bus
    ...

Conceitos

- Características e qualidades de middelware podem incluir:
    - Consistência
    - Redução de Latência
    - Segurança
    - Continuidade e resiliência
    - Abstração e formatação



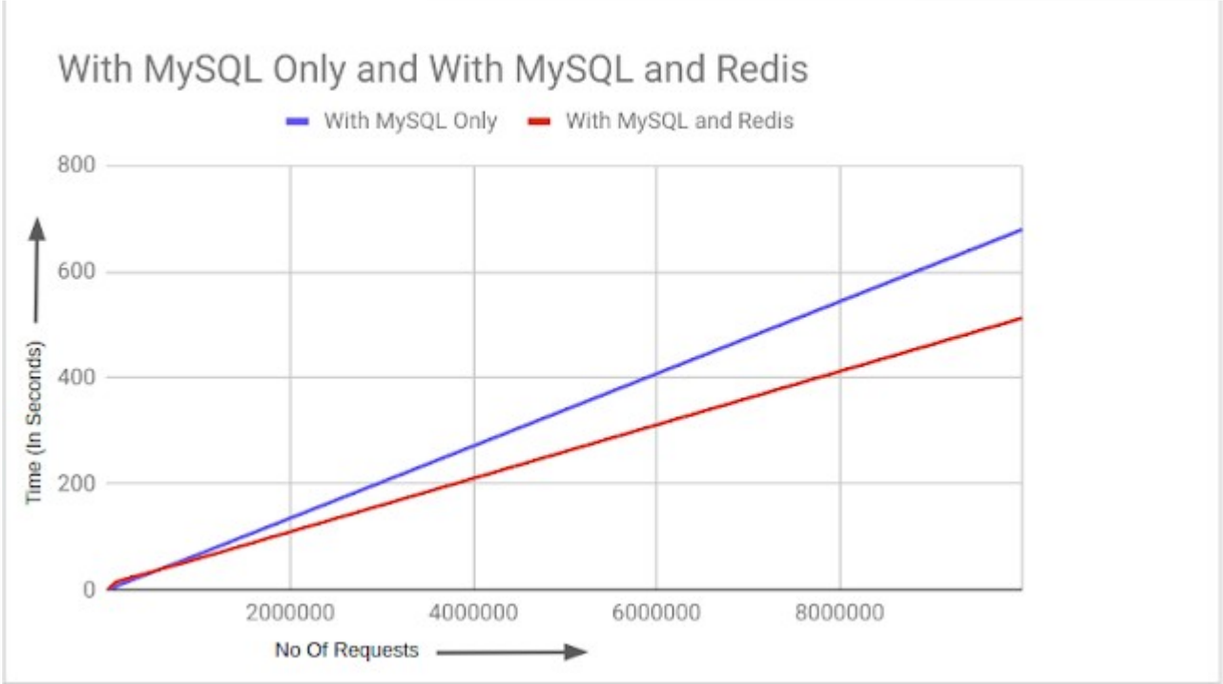
Cache em memória

- Uso de memória RAM para aumentar velocidade no acesso.

- Simples e pragmático.

- Escopo focado.

- Exemplos: memcached, redis, etc



https://dzone.com/articles/redis-vs-mysql-benchmarks.

Exemplo com Redis

- Uso de instância única

Server
#> docker run --name some-redis -d redis redis-server --save 60 1 --loglevel warning

#> docker run --name myredis -d -p 6379:6379 -v /tmp/redisdata/:data redis redis-server --save 60 1 --loglevel warning

Client
#> redis-cli -h localhost -p 6379

Main commands:

set
get
getset
rename
exists
keys
del
ttl

Exemplo com Redis

- Tipos de dados em redis

#string
SET user:1000 "Rootinha"
SET session:client01 "games_profile" EX 3600
SET session:client01 "games_profile" NX

#hash
HSET user:1000 name "Rodrigo" age 47 email "rzcarvalho@ualg.pt"
HGETALL user:1000
HGET user:1000 e-mail

#JSON
JSON.SET user:1000 $ '{"name":"Rodrigo","age":47,"address":{"city":"Nuremberg"}}'
JSON.GET user:1000 .name
JSON.GET user:1000

Muitos outros tipos são aceites: Stream, Bitmaps, SortedSets, Sets, Lists, etc.
#> https://phoenixnap.com/kb/redis-data-types-with-commands

Atividade com Redis

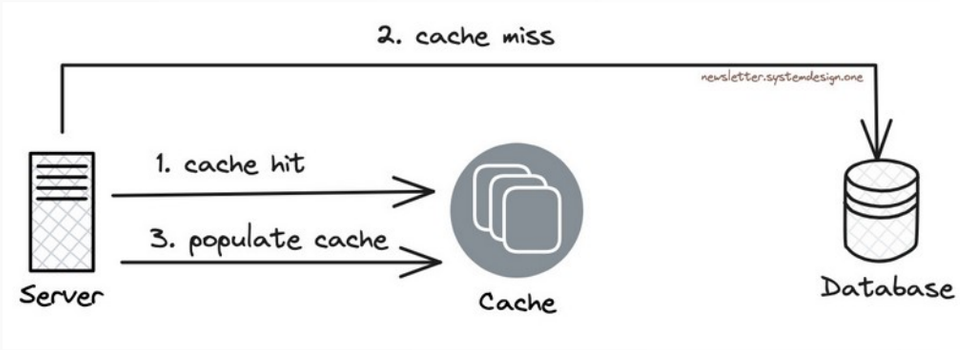
- Crie um servidor redis no seu sistema local e conecte-se com redis cliente.
Em seguida, introduza 3 chaves, de diferentes tipos: string, hash e json.
Certifique-se de que as chaves estão salvas.
Altere o TTL de uma das chaves e confirme que a mesma é removida do cache após sua expiração.

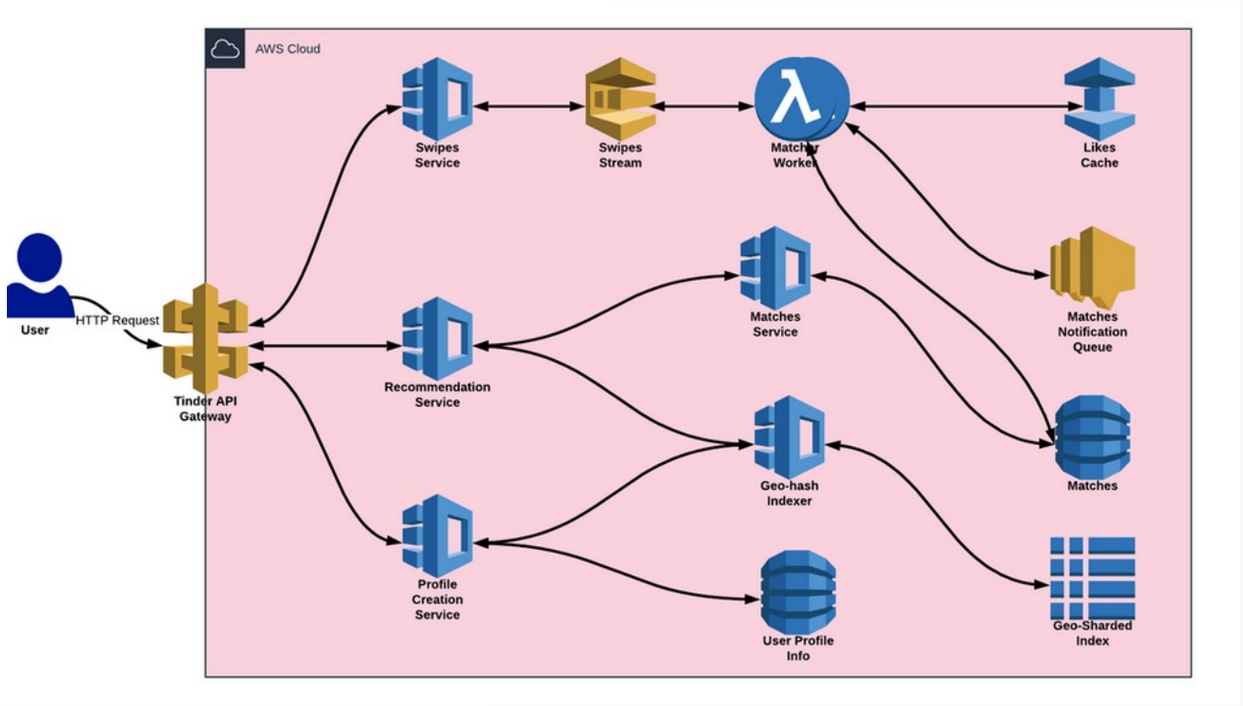
P.S.: Faça uso do https://labs.play-with-docker.com/ se não tem docker instalado no sistema local

Exemplo com Redis

Caso real de sucesso

"...Se não fosse por conta do Redis, eu não estaria casado hoje..."
O uso de Redis na arquitetura do Tinder!





https://newsletter.systemdesign.one/p/tinder-architecture
https://xie.infoq.cn/article/b717a40affb80efa6f9cfee77

Exemplo com Redis

- Uso de cluster com docker-compose.yaml

```
version: '3.8'

services:
  redis-node1:
    image: redis:latest
    container_name: redis-node1
    ports:
      - "7001:6379"
    command: redis-server --cluster-enabled yes --cluster-config-file nodes-node-1.conf --cluster-node-timeout 5000 --appendonly yes
    volumes:
      - redis-data-node1:/data

  redis-node2:
    image: redis:latest
    container_name: redis-node2
    ports:
      - "7002:6379"
    command: redis-server --cluster-enabled yes --cluster-config-file nodes-node-2.conf --cluster-node-timeout 5000 --appendonly yes
    volumes:
      - redis-data-node2:/data

  redis-node3:
    image: redis:latest
    container_name: redis-node3
    ports:
      - "7003:6379"
    command: redis-server --cluster-enabled yes --cluster-config-file nodes-node-3.conf --cluster-node-timeout 5000 --appendonly yes
    volumes:
      - redis-data-node3:/data

volumes:
  redis-data-node1:
  redis-data-node2:
  redis-data-node3:
```

Inicializar o cluster

```
#> docker exec -it redis-node1 redis-cli --cluster create redis-node1:6379 redis-node2:6379 redis-node3:6379 --cluster-replicas 0
```

Broker system

- Termo começa a ser utilizado nos fins dos anos 70s.

- Sistema para interfacear a comunicação ou transação em SD.

- Considerado como sistema intermediário. Isto é; dados são geralmente transientes e server para promover um objetivo maior.

- Permite desacoplar componentes, tornando-os mais independentes.

- Promove escalabilidade.

- Permite modos de comunicação mais flexíveis (Request-Response, Pub-Sub, Filas, ec).

- Message brokers, service brokers e event brokers.

Exemplo com Rabbitmq

- Executando com nó único

- Portas 15672 e 5672

#> docker run -d --hostname my-rabbit --name rabbit13 -p 8080:15672 -p 5672:5672 -p 25676:25676 rabbitmq:3-management

https://medium.com/xp-inc/rabbitmq-com-docker-conhecendo-o-admin-cc81f3f6ac3b

Atividade com Rabbitmq

- *Crie um servidor de filas RabbitMQ. Use o docker para isso.*
  Adicione uma fila e uma mensagem de teste.
  Em seguida faça a leitura (e consumo) dessa mesma mensagem através do comando curl.

P.S.: Faça uso do https://labs.play-with-docker.com/ se não tem docker instalado no sistema local

Exemplo com Rabbitmq

- Executando em cluster

```
version: '3.8'

services:
  rabbitmq1:
    image: rabbitmq:3.11-management
    container_name: rabbitmq1
    hostname: rabbitmq1
    ports:
      - "15672:15672" # Management UI
      - "5672:5672"   # AMQP
    environment:
      - RABBITMQ_ERLANG_COOKIE=SECRETCOOKIE
      - RABBITMQ_DEFAULT_USER=admin
      - RABBITMQ_DEFAULT_PASS=admin123
    volumes:
      - rabbitmq1-data:/var/lib/rabbitmq
    networks:
      - rabbitmq-cluster

  rabbitmq2:
    image: rabbitmq:3.11-management
    container_name: rabbitmq2
    hostname: rabbitmq2
    depends_on:
      - rabbitmq1
    environment:
      - RABBITMQ_ERLANG_COOKIE=SECRETCOOKIE
      - RABBITMQ_DEFAULT_USER=admin
      - RABBITMQ_DEFAULT_PASS=admin123
    volumes:
      - rabbitmq2-data:/var/lib/rabbitmq
    networks:
      - rabbitmq-cluster
    command: bash -c "sleep 10 && rabbitmq-server"
.
.
.

volumes:
  rabbitmq1-data:
  rabbitmq2-data:
  rabbitmq3-data:

networks:
  rabbitmq-cluster:
    driver: bridge
```

Conclusões

- Segurança, criptografia e autenticação também podem (e devem) ser empregues.

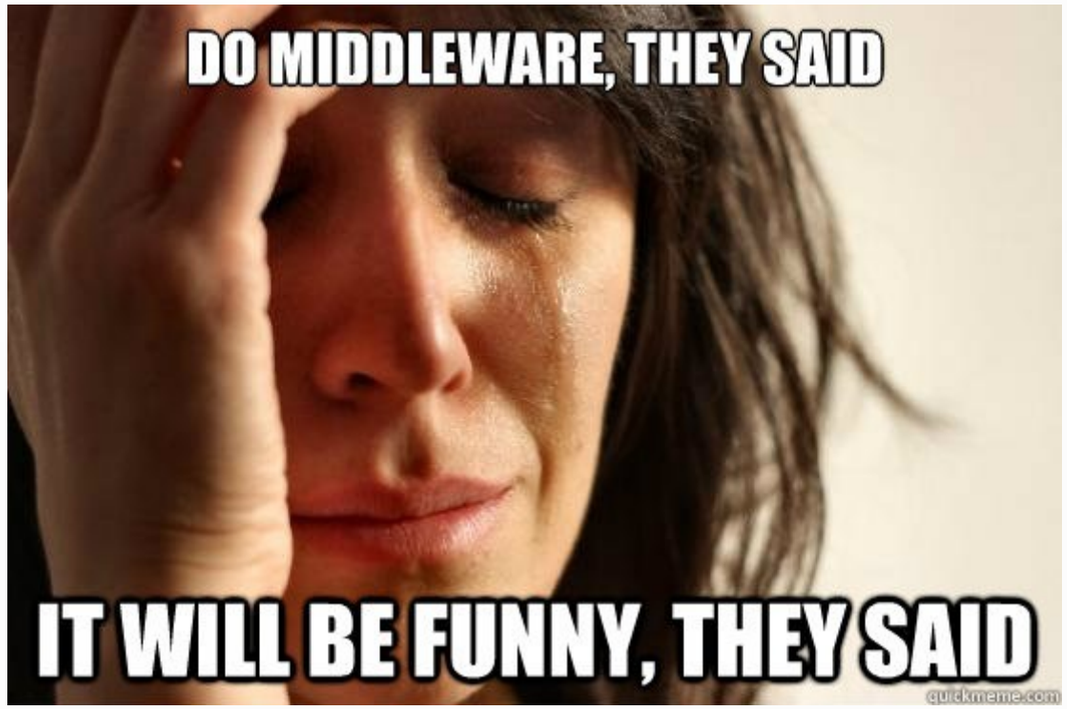
- A distribuição da carga entre nós dos clusters em geral fica a cargo da rede em um SD.

- Sistemas com mais ou menos facilidades na integração, e.g.: cluster redis.

- Outros sitemas a serem utilizados no mesmo contexto: Kafka, Spark, ElasticSearch, etc.

- Middleware mais sofisticados e com escopo mais amplo: Zapier, Workato, Tray.ai, MSPower Automate, etc.

FIM


DO MIDDLEWARE, THEY SAID
IT WILL BE FUNNY, THEY SAID
quickmeme.com