ANÁLISE DE DESEMPENHO COMPUTACIONAL EM MICROSSERVIÇOS GERENCIADOS COM KUBERNETES[1]

Matheus Baltasar de Carvalho Cercena[2], Maurício Aronne Pillon[3].

[1] Vinculado ao projeto "Técnicas de escalonamento, precificação e impacto energético em plataformas distribuídas "
[2] Acadêmico (a) do Curso de Ciência da Computação – CCT – Bolsista PROBIC
[3] Orientador, Departamento de Ciência da Computação – CCT – mauricio.pillon@udesc.br

Microsserviços é uma abordagem em modularização de aplicações distribuídas, cuja característica é dividir funcionalidades de um sistema complexo em módulos menores, a fim de facilitar o desenvolvimento, implementação e entendimento dessa aplicação. No que tange a implementação, o grau de modularização impacta na escalabilidade do microsserviço e no desempenho. O desenvolvimento e modelagem do sistema são afetados na forma de organização de equipes de projetos, coordenação de pessoal e habilidades específicas de cada módulo. Microsserviços prezam pela independência conceitual dos módulos, mas não fogem da comunicação intermódulos por meio de transações de redes.

Para essa arquitetura ser passível de operação, exige-se o estabelecimento de um ambiente distribuído, a qual múltiplos computadores estão dispostos dentro de um mesmo espaço de rede, de maneira que eles consigam se comunicar entre si. O processamento da aplicação é dividido entre os processos pertencentes ao grupo. Ferramentas de conteinerização são úteis em tal cenário, já que elas podem exercer a função de encapsular os módulos da aplicação que estão espalhados dentro do grupo de computadores e, por meio de um orquestrador de contêineres, distribuir os serviços em contêineres aplicando regras de balanceamento de carga, escalonamento, políticas de segurança, entre outras.

Neste trabalho científico, um estudo de caso foi usado como referência para investigação comportamental de um sistema projetado para operar como uma arquitetura de microsserviços. Trata-se de uma aplicação web que simula o ambiente de uma loja virtual, intitulada *SockShop*, contendo diversos módulos separados que fornecem os serviços necessários para seu funcionamento, *i.e.*, banco de dados. Essa aplicação foi implementada na plataforma de conteinerização *Docker*, a qual garante o lançamento dos módulos na rede, e a partir do *Docker*, ser gerenciado com o orquestrador de contêineres *Kubernetes*, que administra e abstrai todos os nós de processamento. Para monitoração, utilizou-se *Prometheus*, ferramenta que coleta métrica da aplicação em tempo de execução, e um gerador de carga artificial, *Locust*, que pode ser aplicado diretamente no sistema para simulação de tráfego.

O fluxo de redes presente no *Kubernetes* exerce uma influência primordial no redirecionamento de pacotes entre as entidades da arquitetura (interfaces de rede, roteadores, contêineres), estabelecendo regras de encaminhamento e filtragem do conteúdo. A eficiência de consumo dos recursos computacionais presente na plataforma de processamento usada para manter o sistema (processador, memória) depende, entre outros fatores, do quão balanceado a carga está. O volume de tráfego e interações concorrentes que a aplicação está apta a receber também podem influenciar no desempenho e/ou no funcionamento no sistema.

Para a elaboração das análises, foi necessário o planejamento de cenários de testes que potencializam mudanças no estado de operação dos recursos. O balanceamento de carga frente ao acesso à plataforma é uma situação que faz testar os limites operacionais do sistema, a qual se descobre o ponto de tráfego que faz com que a aplicação se torne instável. A mudança

comportamental de um cenário para o outro pode ser verificada comparando os gráficos de desempenho que a ferramenta de monitoramento oferece para os recursos alvo.

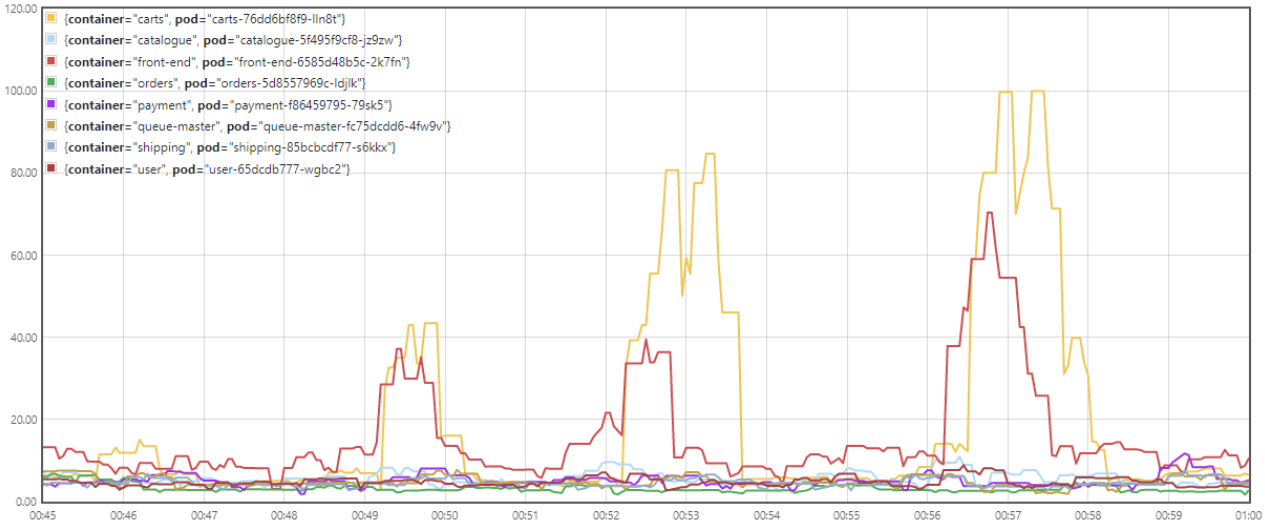

Figura 1. *Gráfico de uso de processador entre os contêineres da aplicação*

O gráfico da figura 1 é resultado de uma *query* executada no *Prometheus* que coleta o uso de *CPU* relativo à quantidade disponível que está alocada para cada um dos contêineres presentes na plataforma, filtrando apenas aqueles que estão relacionados ao funcionamento da aplicação em si. O eixo y representa o uso de *CPU* em porcentagem, o eixo x a hora em tempo real, e a legenda identifica os principais *pods* (entidade base no *Kubernetes*) que compõe cada módulo da aplicação.

Nota-se três grandes picos de uso no tempo capturado que fogem da baixa utilização usual do sistema em menos de 20%, esses momentos foram causados pelo estresse gerado com o balanceador de cargas *Locust***,** que simula usuários artificiais para acessar diretamente o *front-end* da aplicação e assim executar um determinado número de requisições *HTTP* frente às páginas oferecidas pelo *website*. Respectivamente, as execuções do teste foram feitas com os seguintes parâmetros: 10 usuários e 10 requisições, 10 usuários e 50 requisições, 25 usuários e 100 requisições. Os módulos mais impactados foram os de *front-end* e o do ‘carrinho de compras’, este último que ao atingir os 100% de utilização no teste final, falhou em processar requisições restantes devida tamanha incapacidade. Esse experimento evidencia a importância de se definir bem a alocação de recursos ao se lançar contêineres e explorar os limites operacionais deles de forma com que se assegure um funcionamento previsível da aplicação distribuída como um todo. Os resultados evidenciam a importância de sistema adaptáveis de alocação de recursos de acordo com a demanda.

Palavras-chave: Microsserviços. Conteinerização. Monitoramento. Benchmark. Kubernetes.